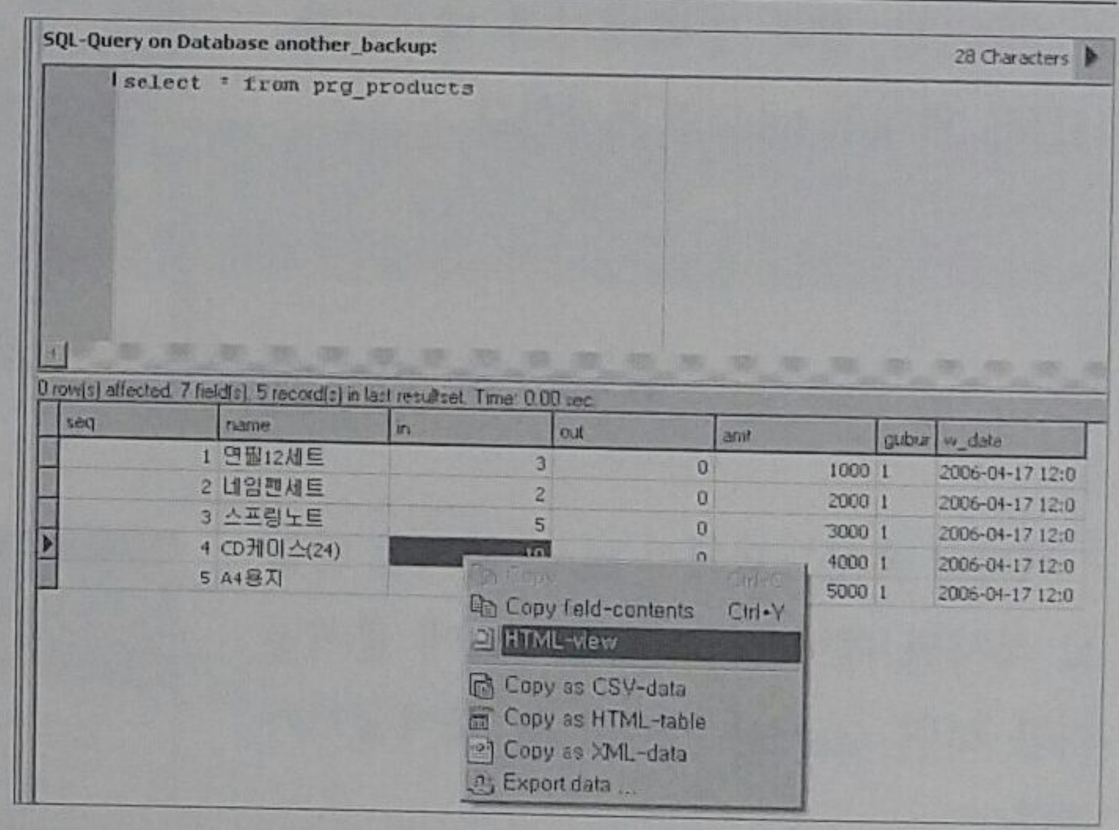
06 Front의 기타 기능과 ODBC 사용하기

안녕하세요 이번 장에서는 Front에 대한 기타 기능에 대해서 공부한다. SQL을 통해서 얻어진 결과를 csv 파일 또는 기타 파일로 저장하는 방법과 작성한 SQL문의 저장 등에 대해서 배우고 Front의 중요 기능 이외의 것들에 대해서도 설명한다.

Front의 기타 기능

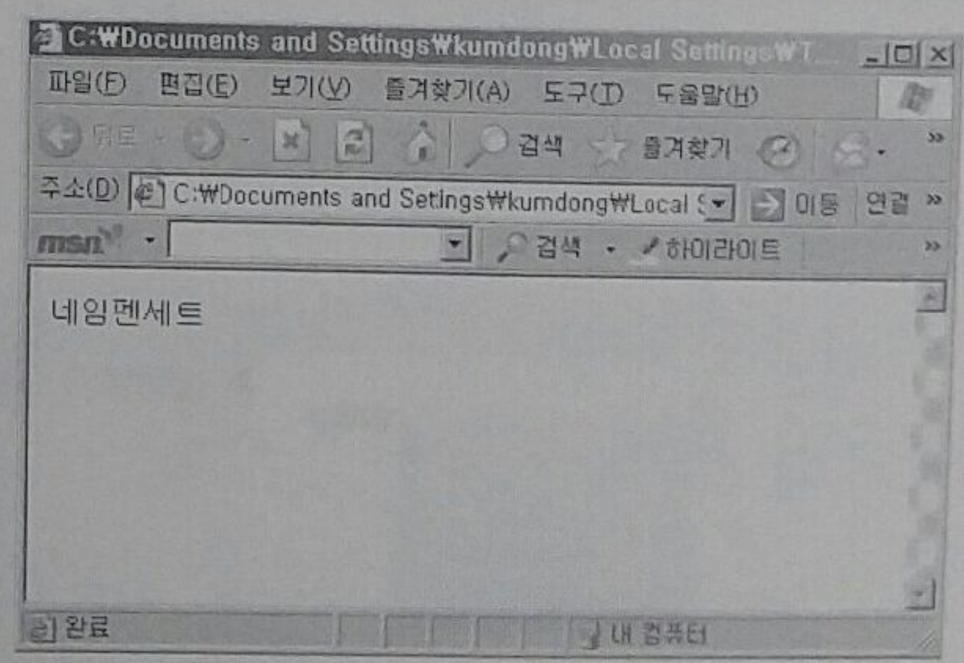
HTML_VIEW

select한 결과 영역에서 마우스의 오른쪽 버튼을 클릭하면 팝업 메뉴가 나온다.



▶ 팝업 메뉴

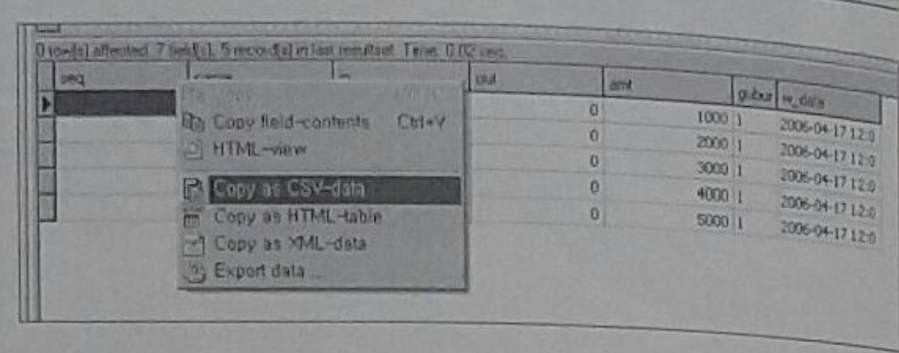
HTML_VIEW를 선택하면 선택된 필드의 값이 HTML 브라우저에 보인다.



▶ 브라우저에 보이는 필드값

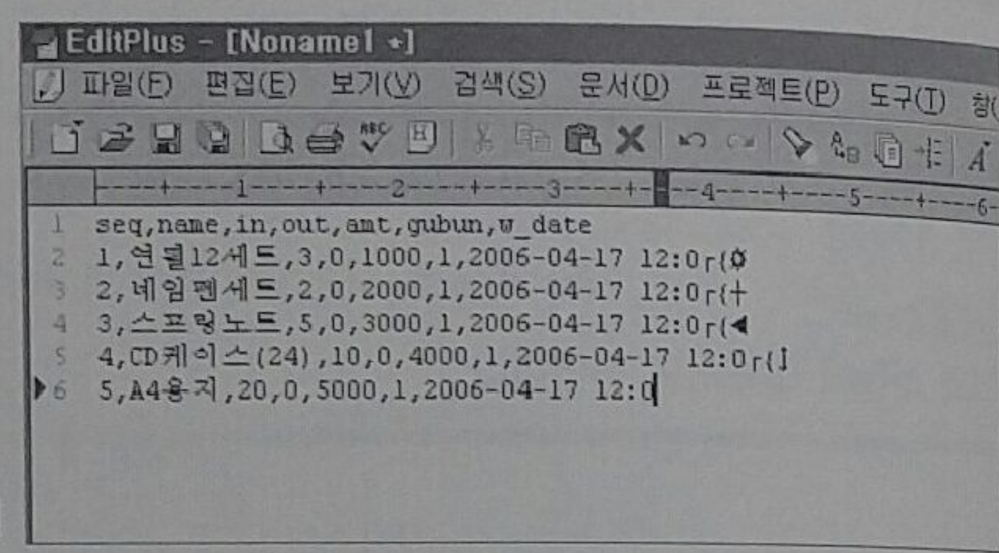
Copy as CSV-data

select한 결과 영역에서의 데이터를 csv 형식으로 복사한다.



▶ Copy as CSV-data

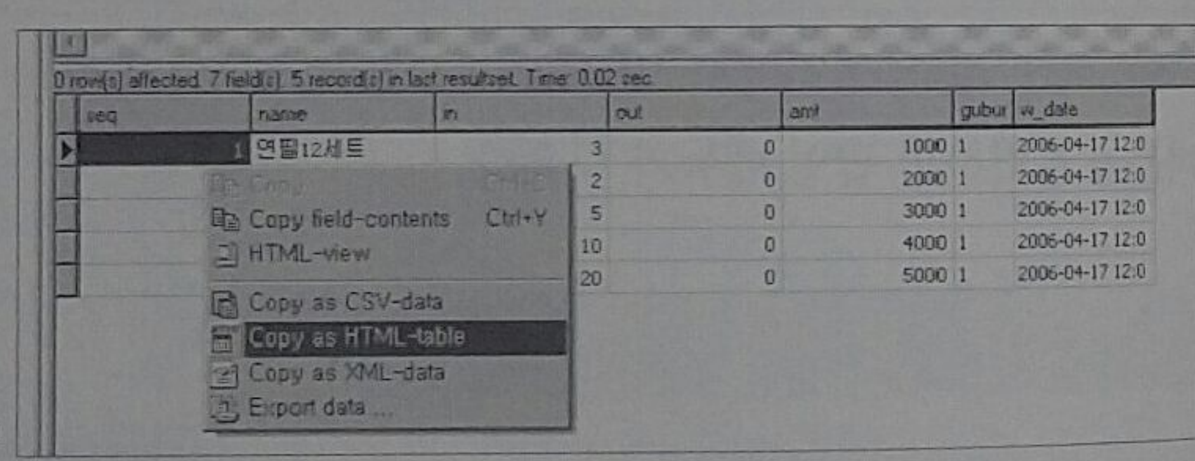
복사한 데이터를 메모장 또는 기타 에디터 프로그램에서 새 문서를 통해 붙여넣기를 하면 데이터가 복사된 것을 확인한다.



▶ 붙여넣기

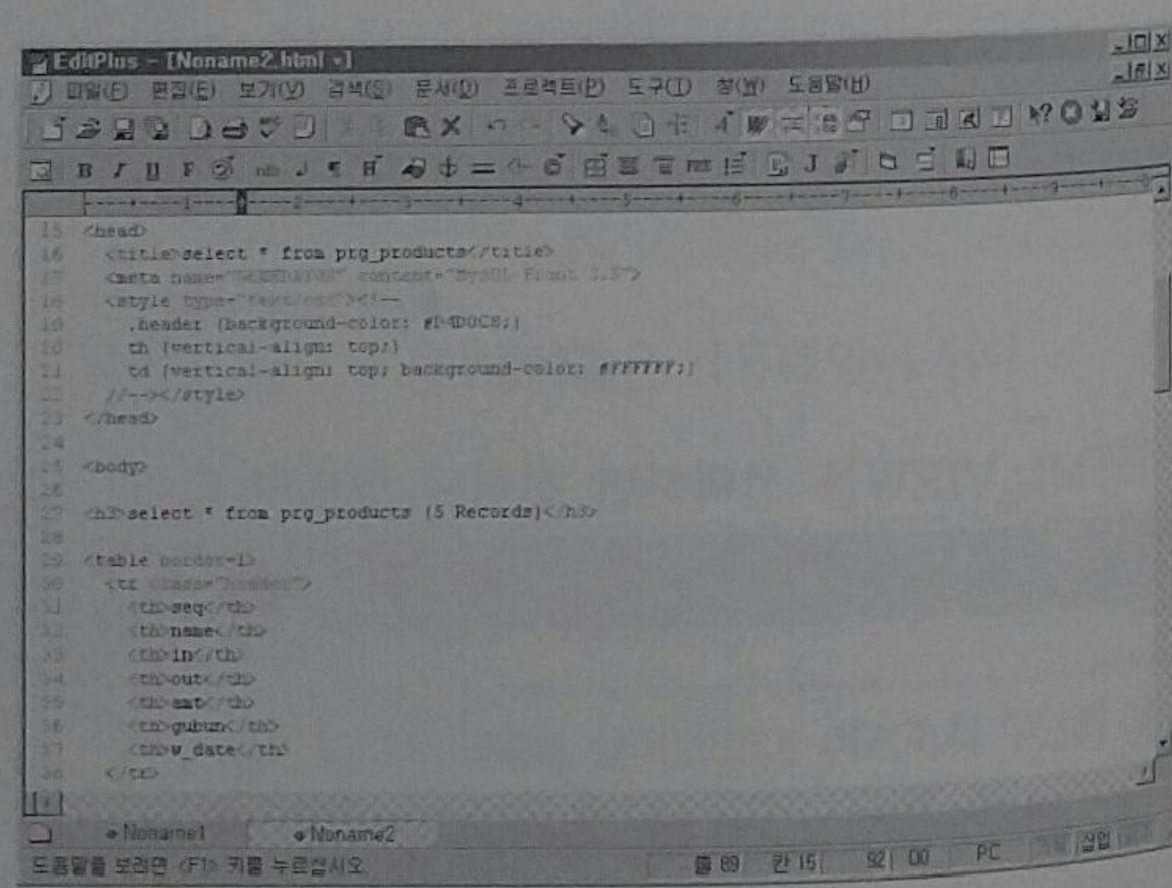
Copy as HTML-table

select한 결과 영력에서의 데이터를 HTML 형식의 table로 복사한다.



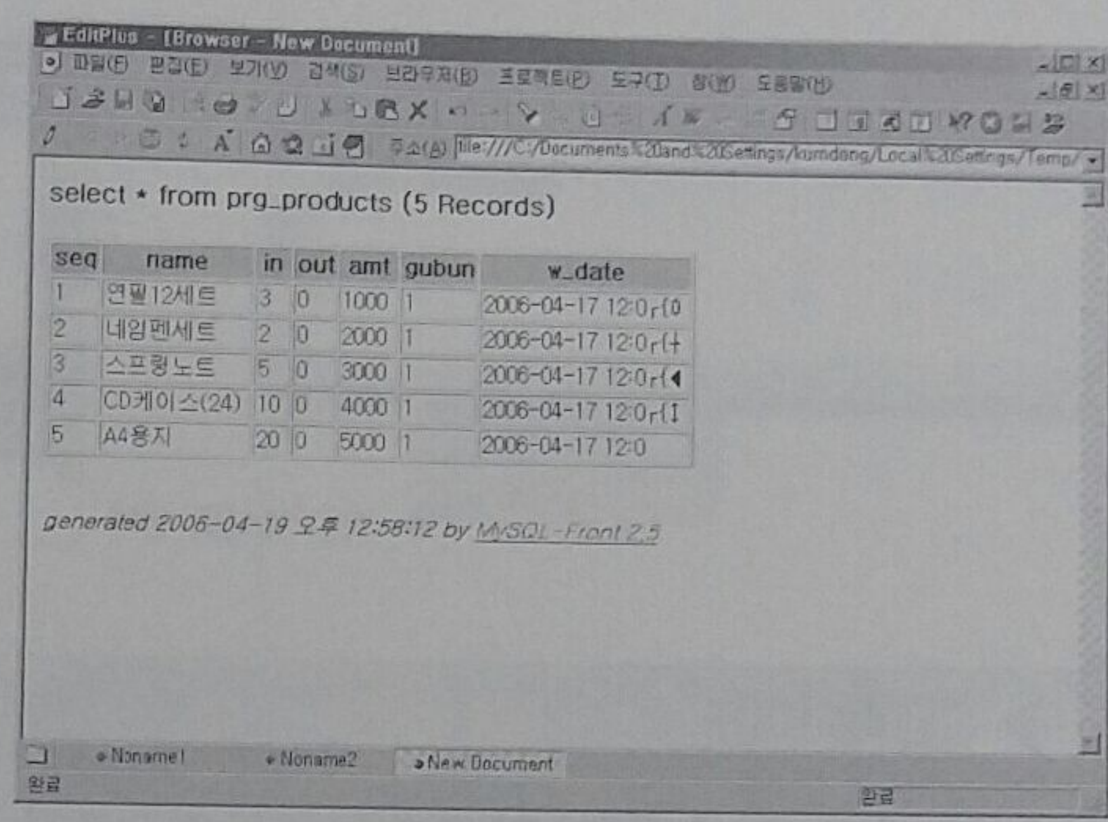
▶ Copy as HTML-table

에디터 프로그램에서 html 문서를 만들고 복사된 데이터를 html 형식에 붙여넣기를 하면 〈table〉 형식으로 붙여넣기가 된다.



▶ 붙여넣기

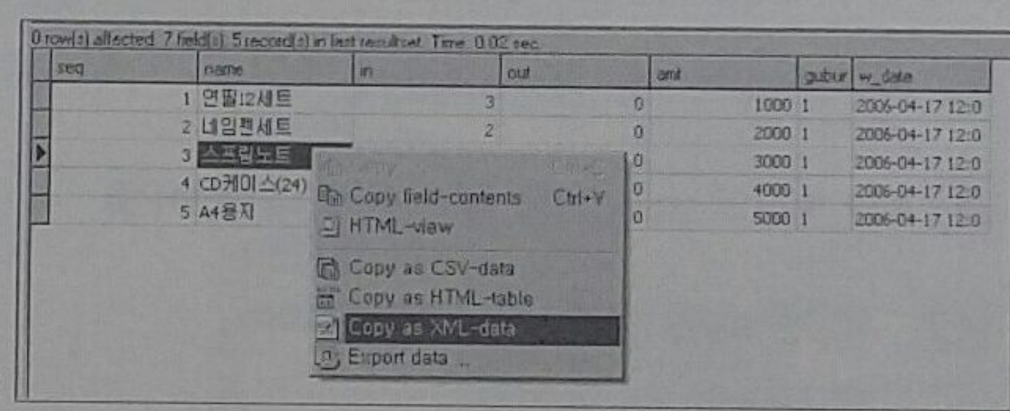
붙여넣기가 된 파일을 저장하여 브라우저에서 보거나 또는 기타 에디터 프로그램에서 미리 보기를 이용하여 확인한다.



▶ 복사확인

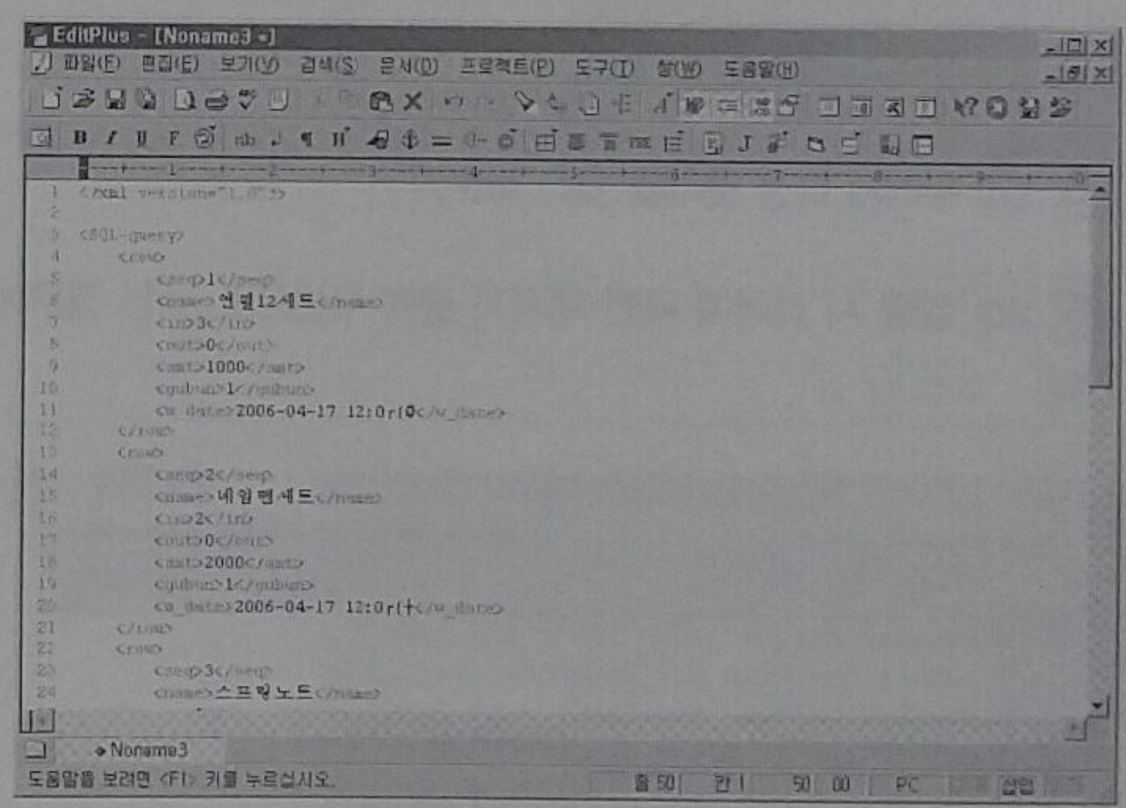
Copy as XML-data

select한 결과 영역에서의 데이터를 XML 형식으로 복사한다.



▶ Copy as XML-data

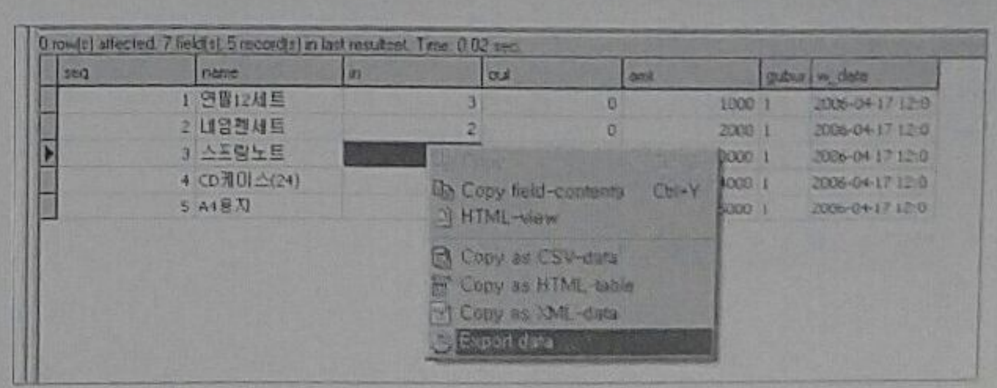
에디터 프로그램에서 XML 문서를 새로 만든 다음 붙여넣기를 하면 XML 형식의 데이터가 만들어지는 것을 확인한다.



▶ XML로 붙여넣기

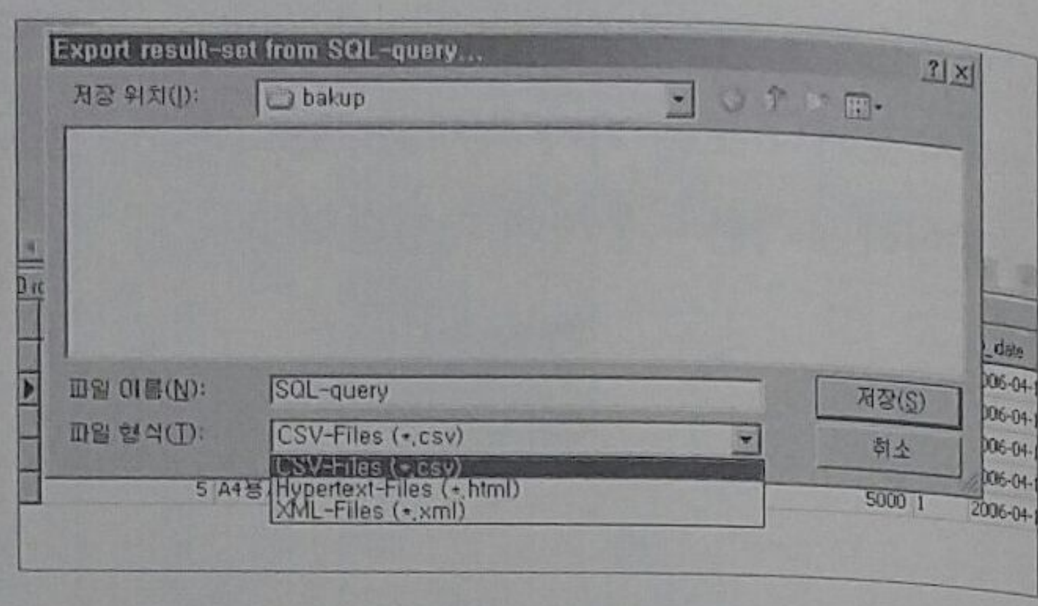
Export

사용자가 csv, html, XML의 파일로 선택하여 저장할 수 있다.



▶ Export

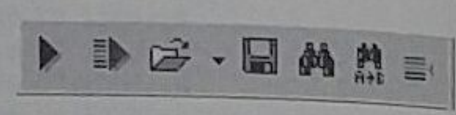
사용자가 데이터를 저장할 파일의 형식(csv, html, XML)을 선택하여 저장할 수 있다.



▶ 파일 형식의 선택

단축아이콘

텝 메뉴의 Query를 선택하면 작성한 sql의 저장 열기 등의 단축 아이콘이 있다.



▶ 단축아이콘

각각의 기능을 살펴보면 다음과 같다.

▶ 하나의 sql이 있을 경우 사용한다.

하나 이상의 sql이 있을 경우 사용한다.(ctrl+F9)

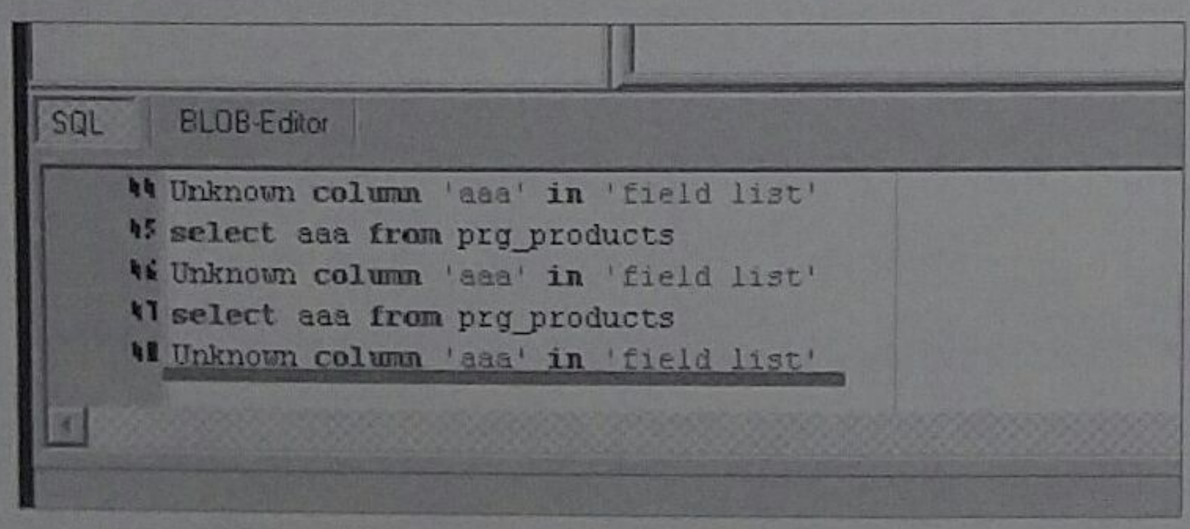
저장된 sql을 열기

작성된 sql 저장하기

sql 문에서 특정 문자 찾기

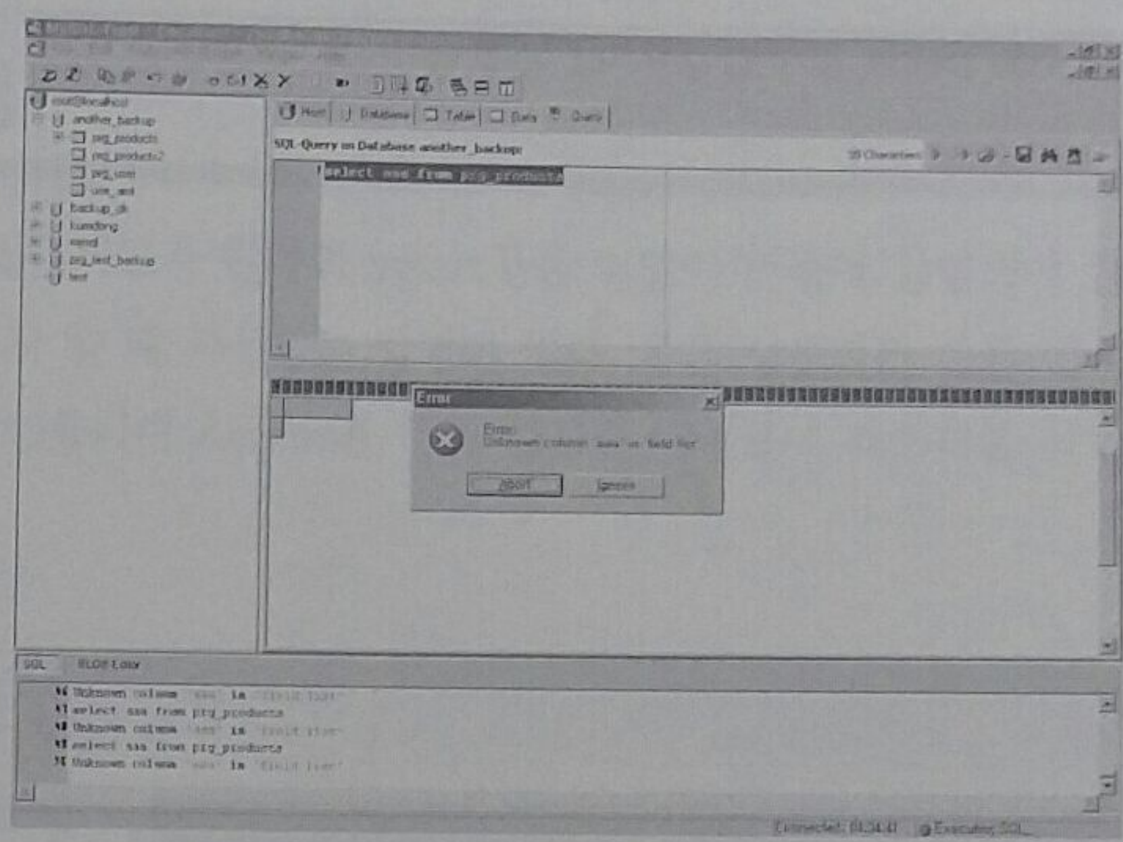
sql 문에서 특정 문자를 찾아 바꾸기

sql 실행 시 에러에 대한 메시지 출력 부분으로 off인 경우에는 에러의 메시지가 sql 상태 영역에 메시지를 출력한다.



▶ off시에 에러메시지

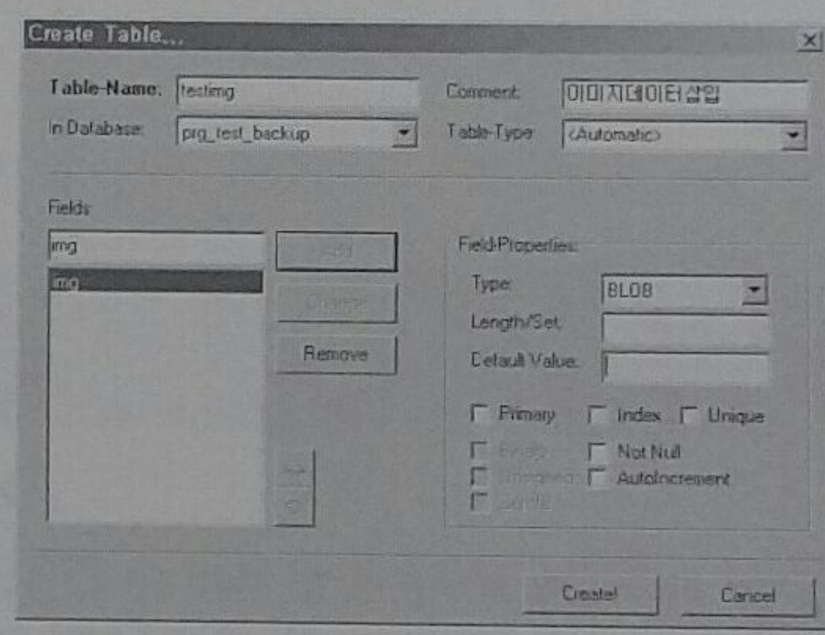
on인 경우에는 메시지 팝업으로 그 에러 상태를 보여준다.



▶ on시에 에러 메시지

이미지 데이터 삽입하기

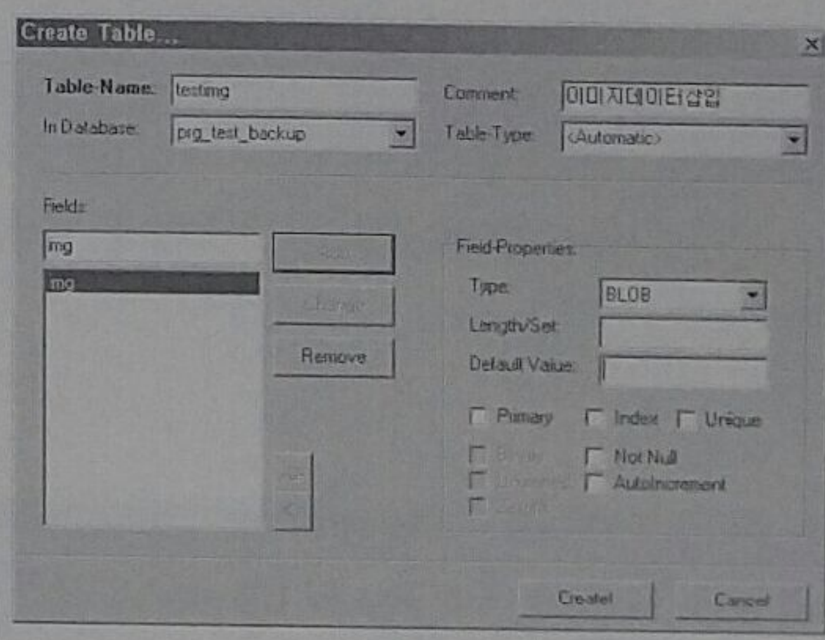
데이터에 이미지 데이터를 저장할 때 사용된다. 하나의 테이블을 생성한다.



▶ 이미지 데이터 저장을 위한 테이블의 생성

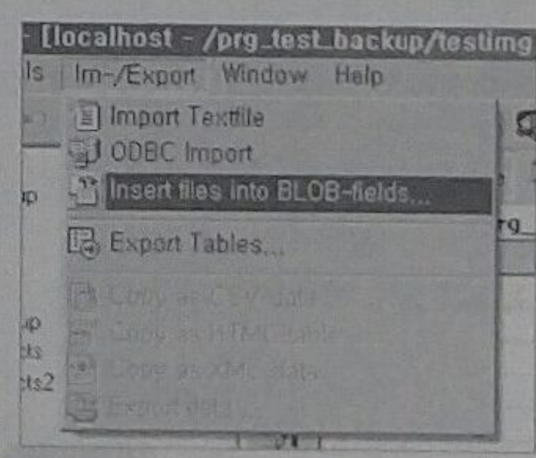
따라하기 이미지 데이터 삽입

❶ Im-/Export 메뉴에서 insert files into BLOB-fields를 선택한다.



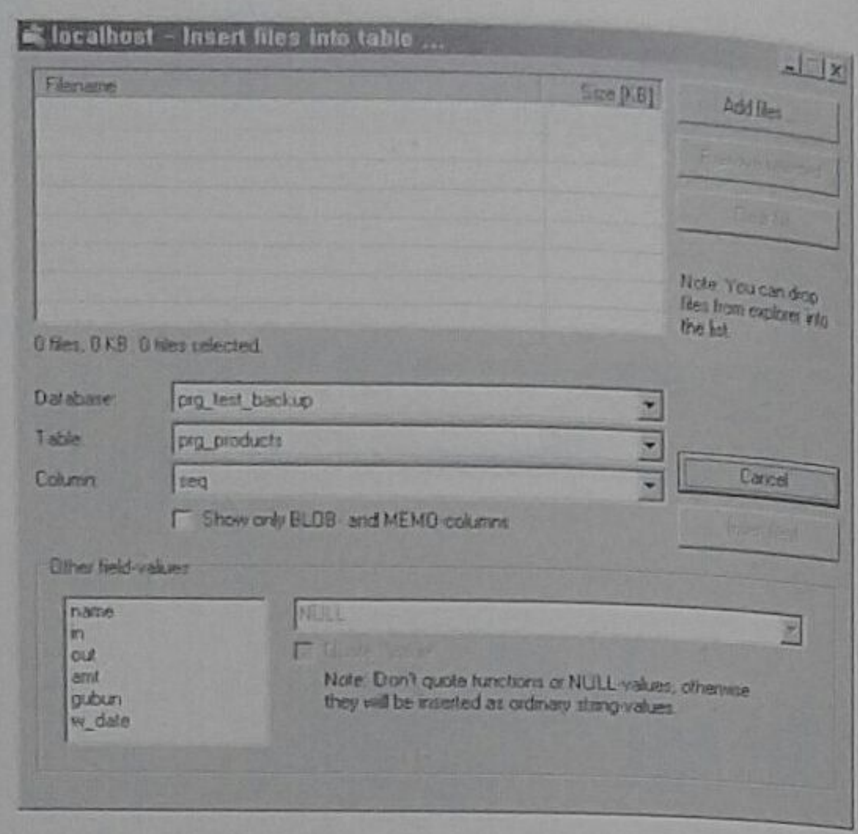
▶ insert files into BLOB-fields 선택

❷ insert files into BLOB-fields에 해당하는 창이 출력된다.



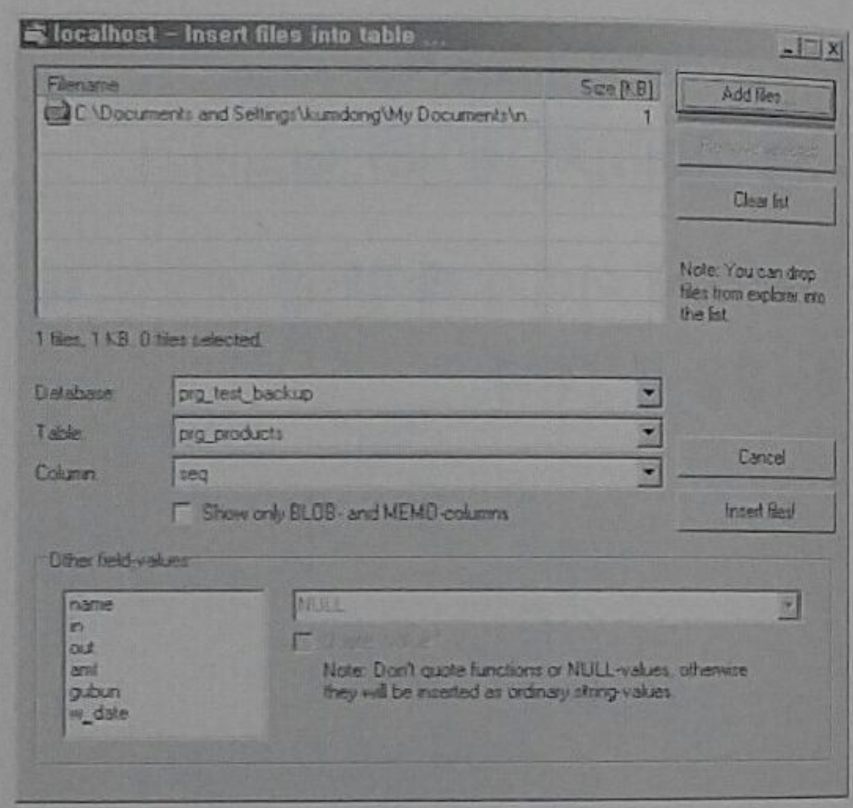
▶ insert files into BLOB-fields 창

❸ Add files를 통해 삽입하고자 하는 이미지를 선택한다.



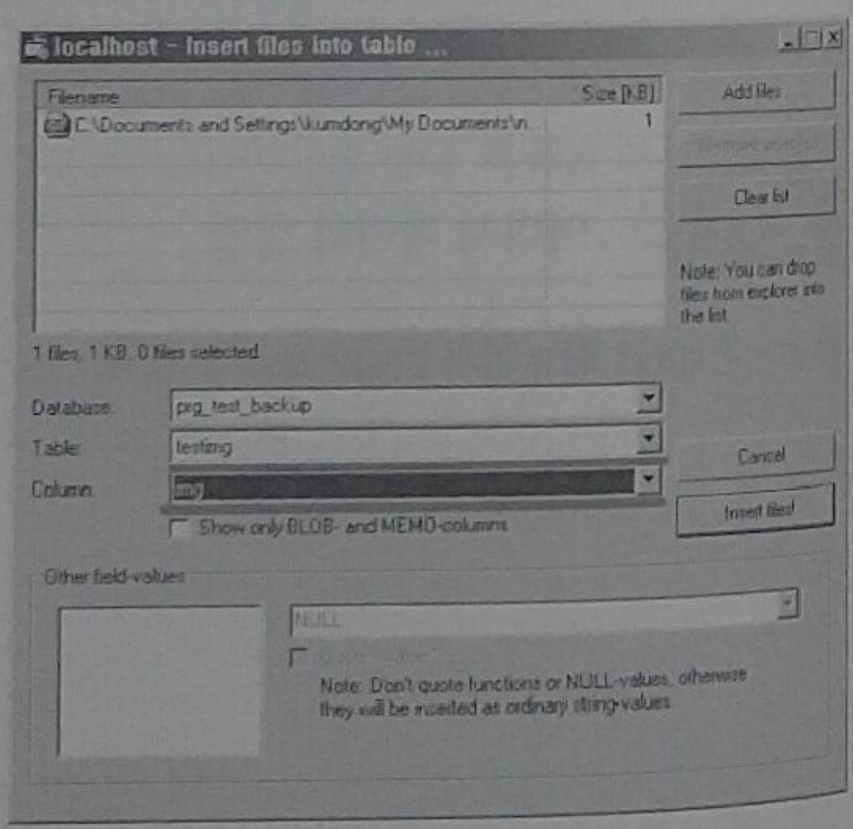
▶ 이미지 선택

❹ 해당 데이터베이스의 테이블과 칼럼을 선택한다.



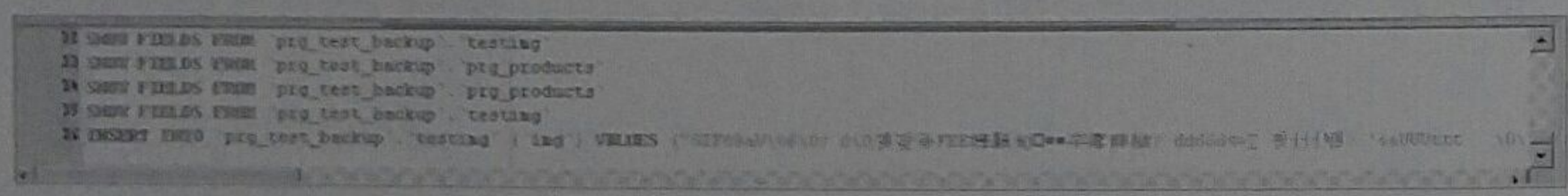
▶ 테이블과 칼럼의 선택

❺ insert files를 클릭하고 나면 메시지 창에 이미지가 insert된 메시지가 출력되는 것을 볼 수 있다.



▶ 메시지

❻ 열려진 insert files into BLOB-fields 창을 닫고 생성한 테이블 선택 후 탭 메뉴의 data를 클릭하면 저장된 이미지 데이터가 보인다.



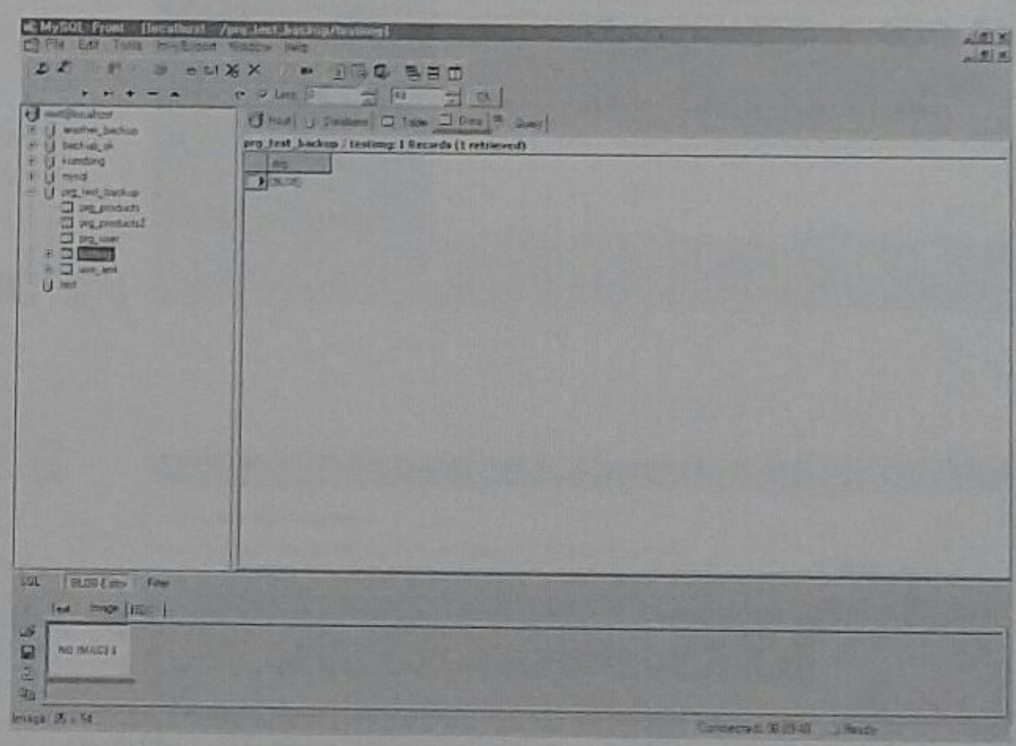
▶ 이미지 데이터의 저장결과

ODBC를 이용한 데이터 작업

많은 양의 데이터를 Mysql에 강제적으로 삽입하고자 한다면 insert문을 일일이 만들어 삽입해야 한다. 그러나 MySql는 그런 데이터를 삽입하고자 할 때 사용하는 방법이 있다. 물론 상용화된 다른 데이터베이스에도 기능이 있다. 여기선 ODBC를 이용하여 Access를 통해 생성된 테이블에 데이터를 삽입할 것이다.

ODBC란?

ODBC 정의라는 검색어를 통해 검색 사이트에서 검색해 보면 ODBC의 정의에 대해서 설명되어 있다.

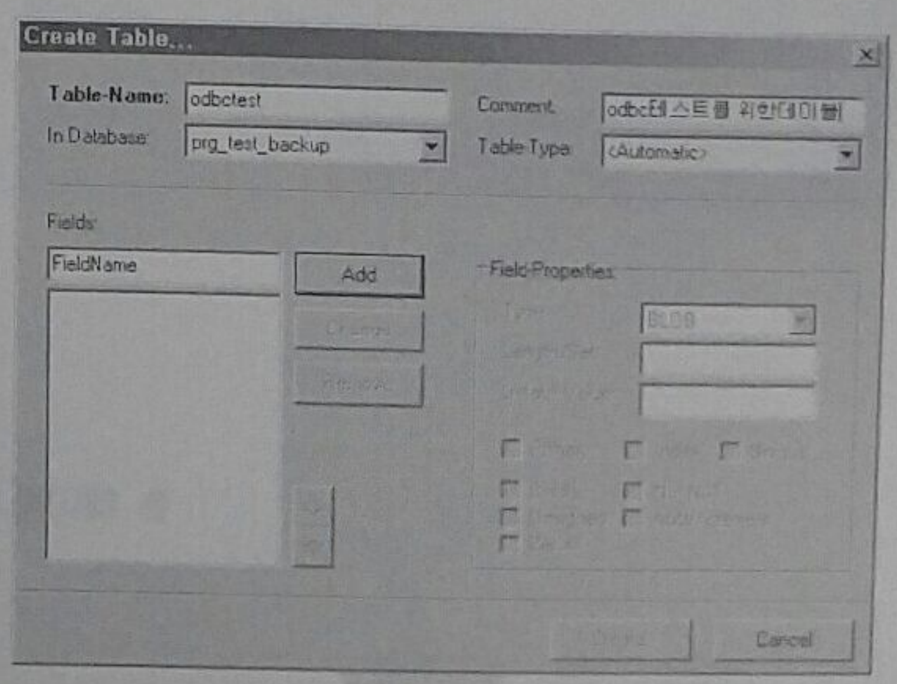


▶ 검색을 통한 ODBC의 정의

좀 더 쉽게 말하면 여러 종류의 데이터베이스와 상관없이 접근할 수 있는 방법이라고 생각하면 된다. 접근 방법은 그 데이터베이스에 접근할 수 있는 드라이버를 먼저 찾아 등록하고 DSN(Database Source Name)를 주고 해당 데이터베이스에 접근할 수 있는 아이디와 패스워드만 주면 손쉽게 해당 데이터베이스에 접근할 수 있다.

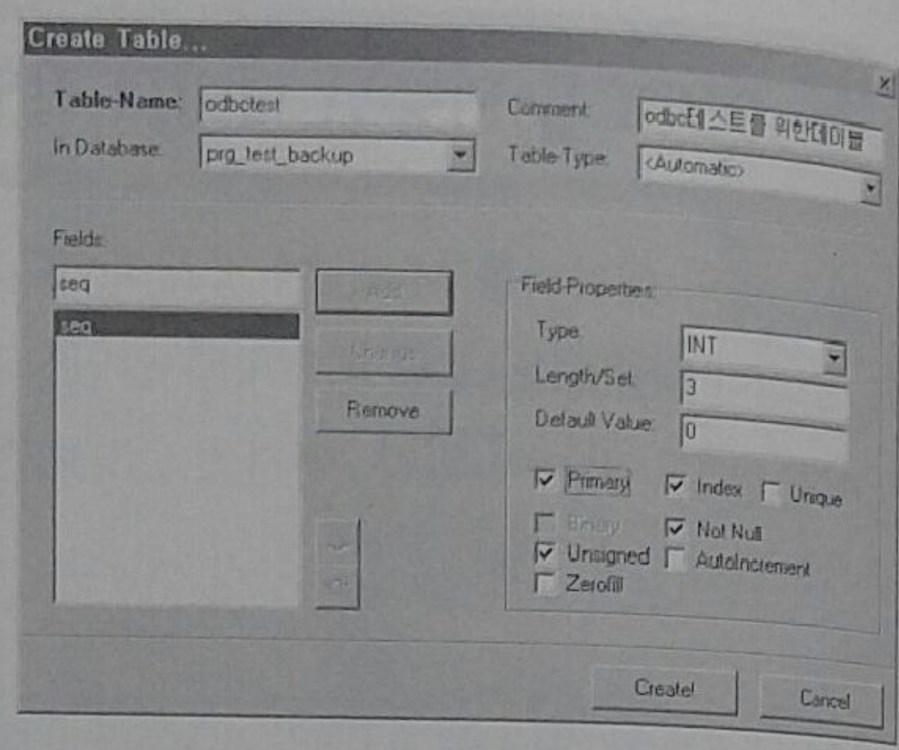
테스트 테이블의 생성

ODBC를 통해 데이터 삽입 과정을 위해 odbctest라는 테이블을 생성하고 각 필드를 생성한다.



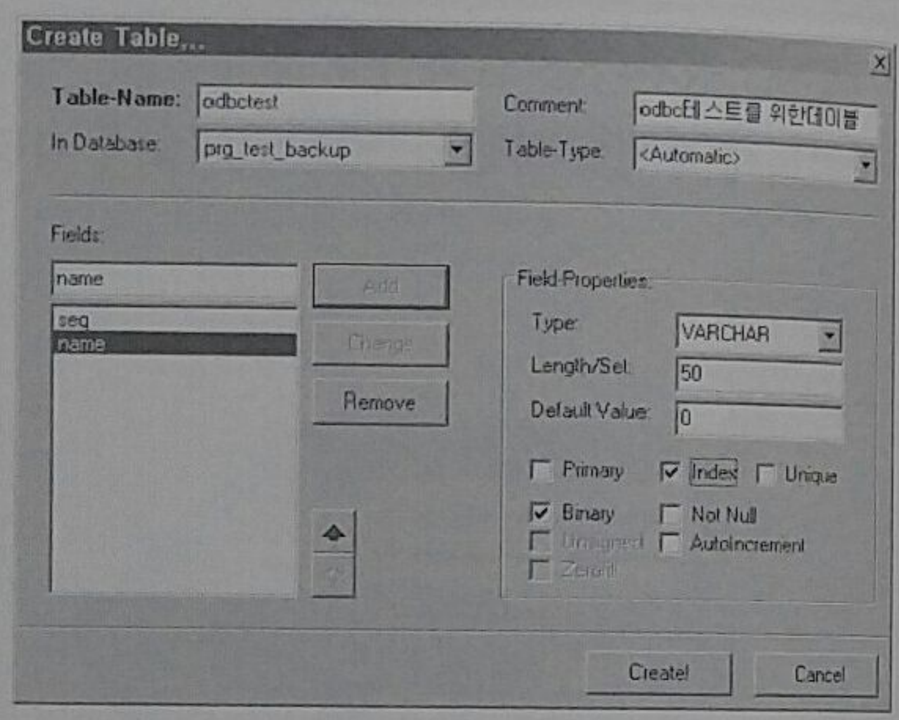
▶ ODBC 테스트 테이블 생성

❷ 일련번호 필드를 생성한다. 단 자동증가 값을 주지 않는다.



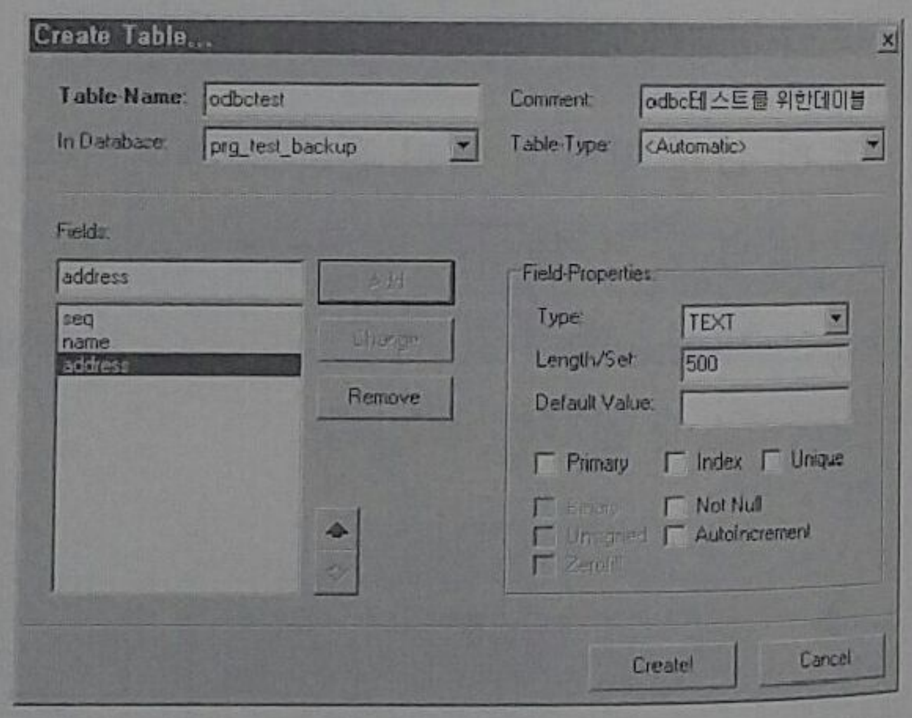
▶ 일련번호 필드 생성

❸ 이름 필드를 생성한다.



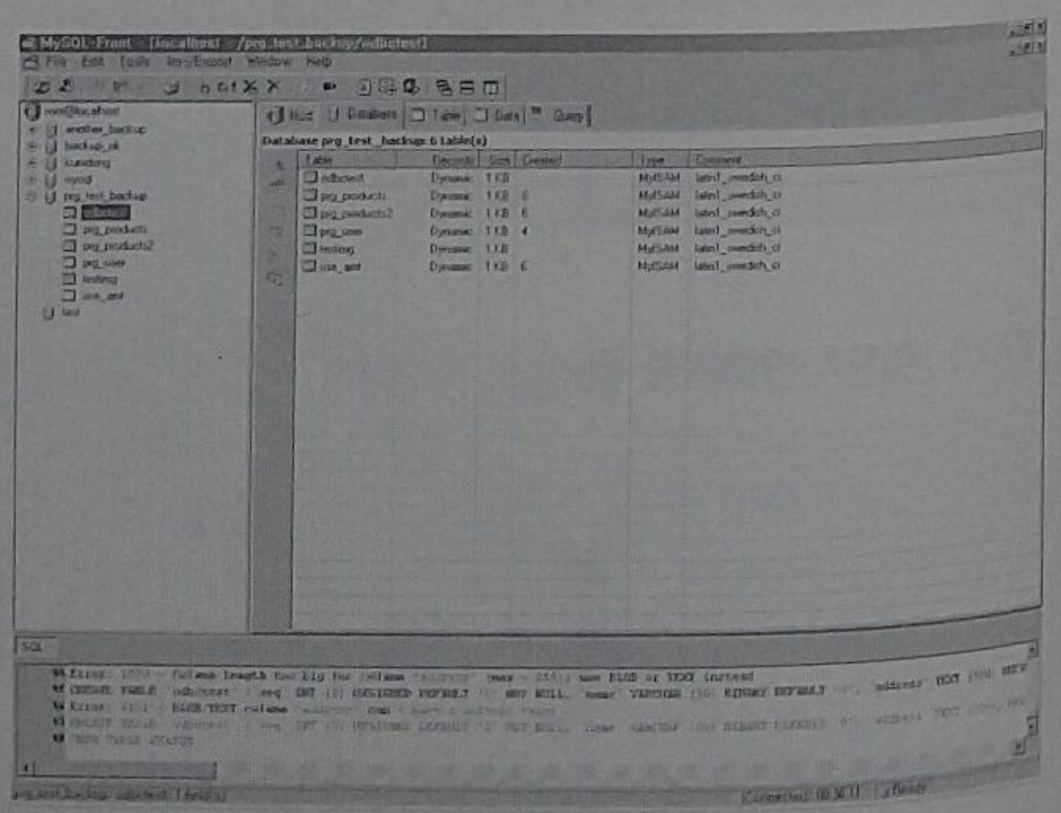
▶ 이름 필드 생성

❹ 주소필드를 생성한다. 디폴트 값이 없음에 유의한다.



▶ 주소 필드 생성

❺ Create 버튼을 클릭하여 테이블을 생성한다.



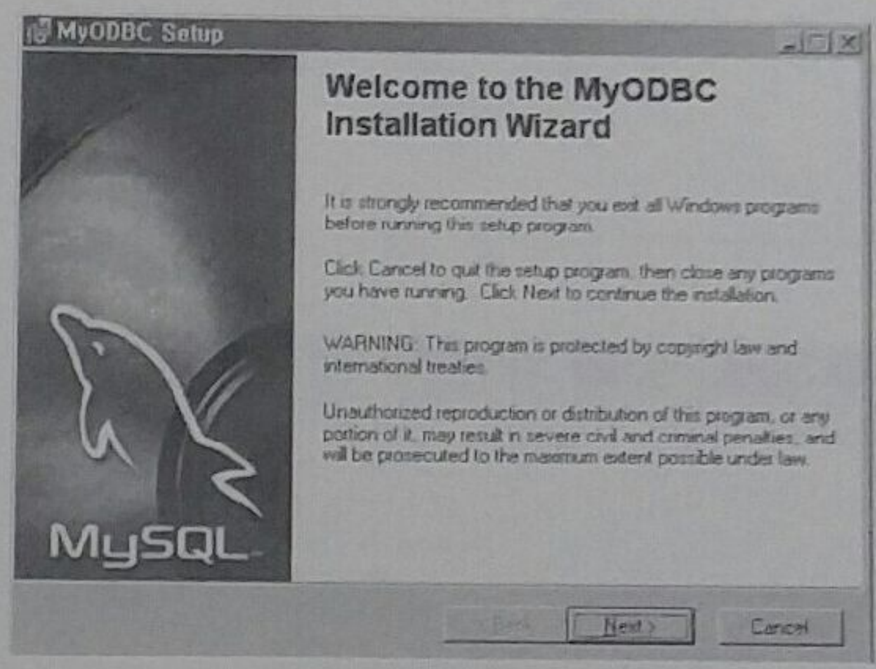
▶ 테이블 생성 확인

MySql ODBC의 설치

ODBC를 위한 테스트 테이블이 생성이 되었다면 MySql용 ODBC를 설치해야 한다.

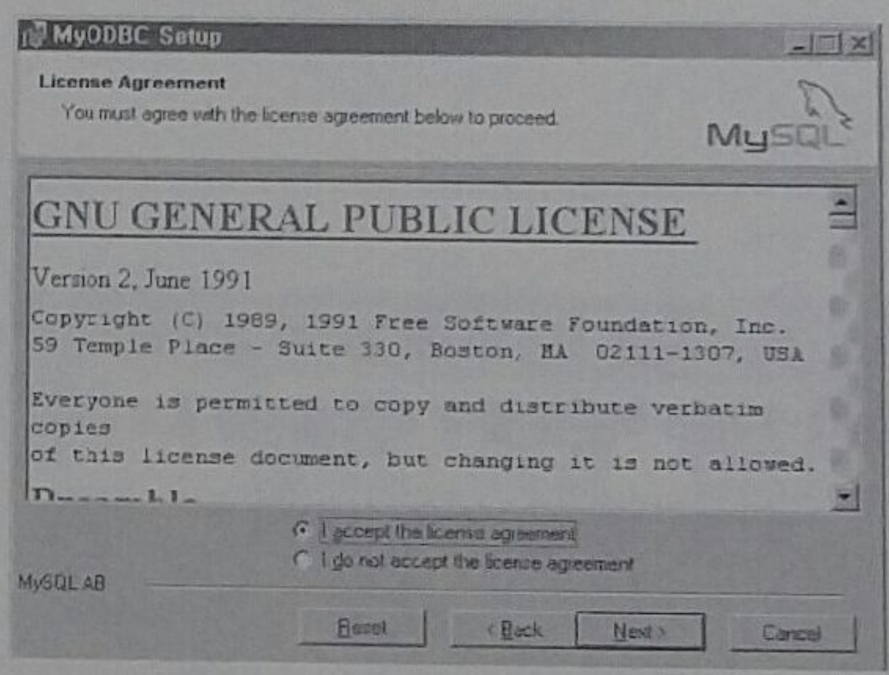
따라하기 MySql ODBC 설치

❶ 대림 홈페이지 자료실에서 다운받은 프로그램자료(mysql_odbc)의 파일을 실행한다.



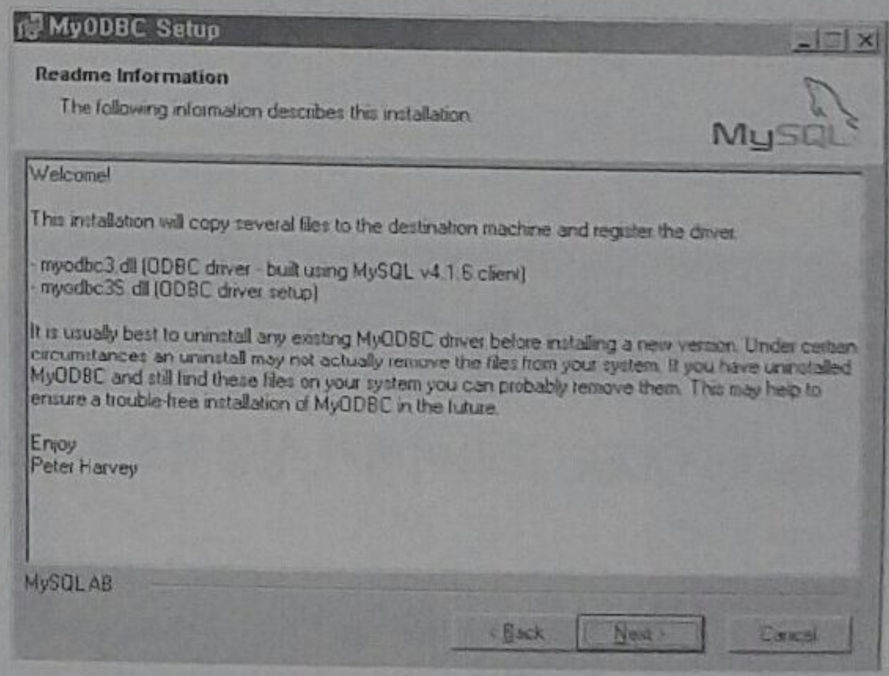
▶ MySql ODBC 설치

❷ next를 클릭하고 라이센스 사용에 동의하고 next 버튼을 클릭한다.



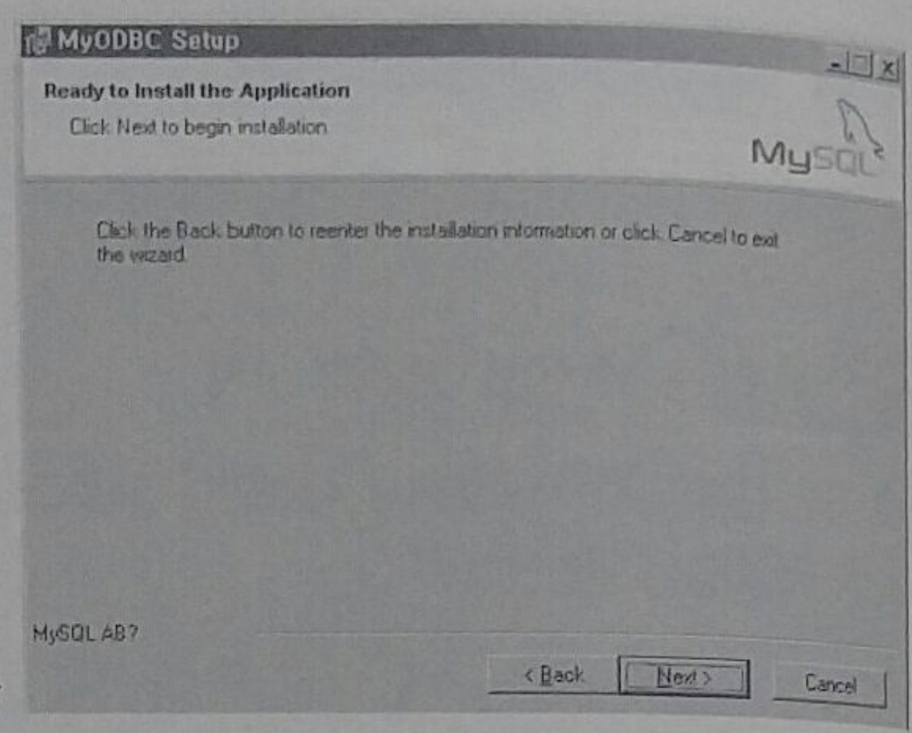
▶ 라이센스 동의

❸ install에 대한 정보가 출력된다. next 버튼을 클릭한다.



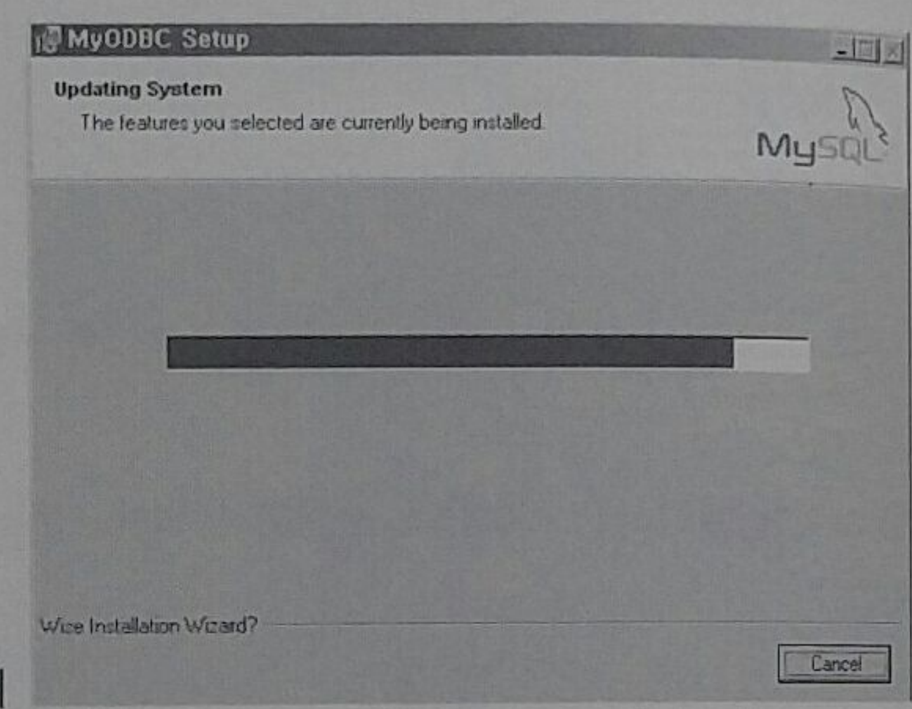
▶ install에 대한 정보 출력

❹ install을 시작할 것인지에 대해 묻는다. next 버튼을 클릭한다.



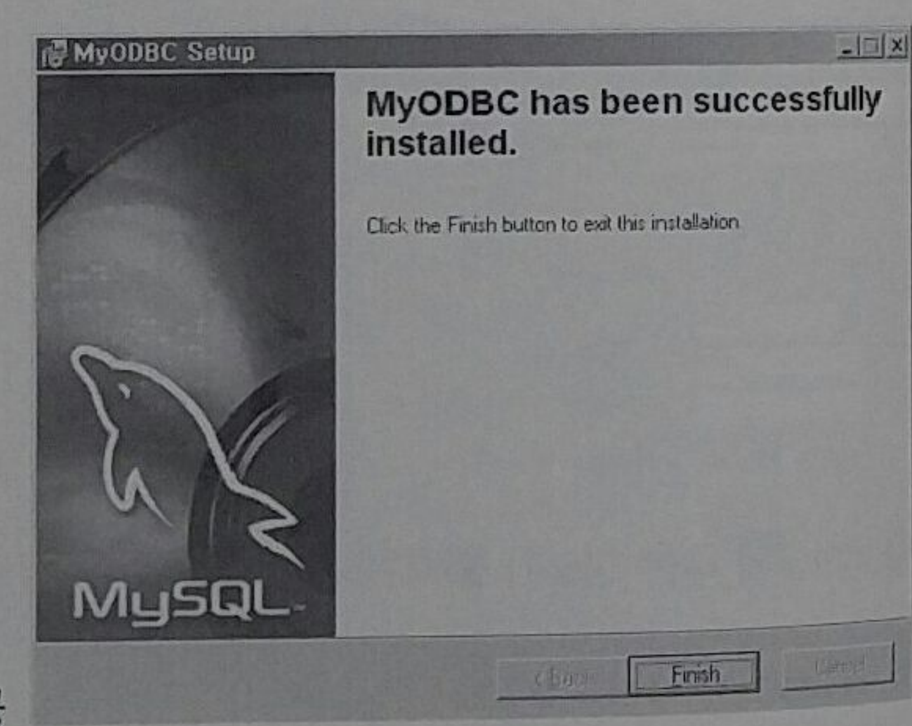
▶ install의 시작

❺ ODBC 설치를 시작한다.



▶ 설치

❻ 설치가 되면 성공적으로 설치가 완료되었다는 메시지를 출력한다. Finish 버튼을 클릭 하여 설치를 종료한다.



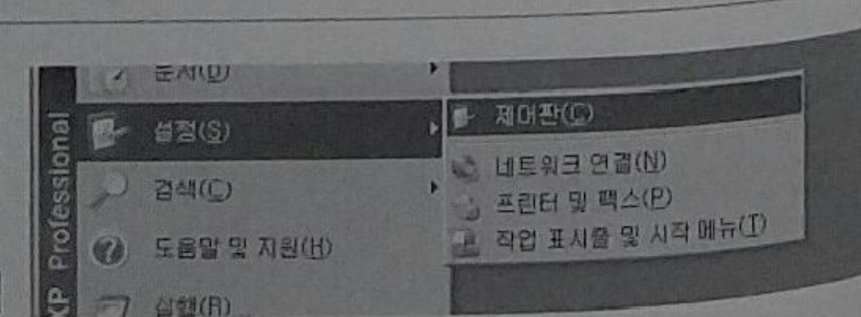
▶ 설치성공

DSN 등록

MySql ODBC 드라이버가 성공적으로 설치가 되었다면 이제 윈도우에서 DSN를 등록해 본다.

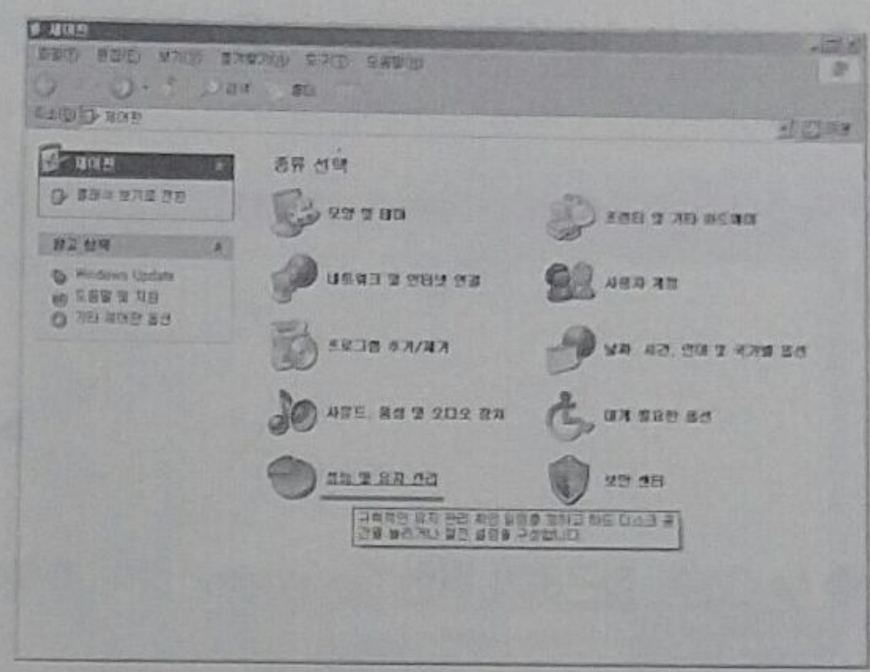
DSN 등록

❶ 윈도우 시작 → 제어판을 선택한다.



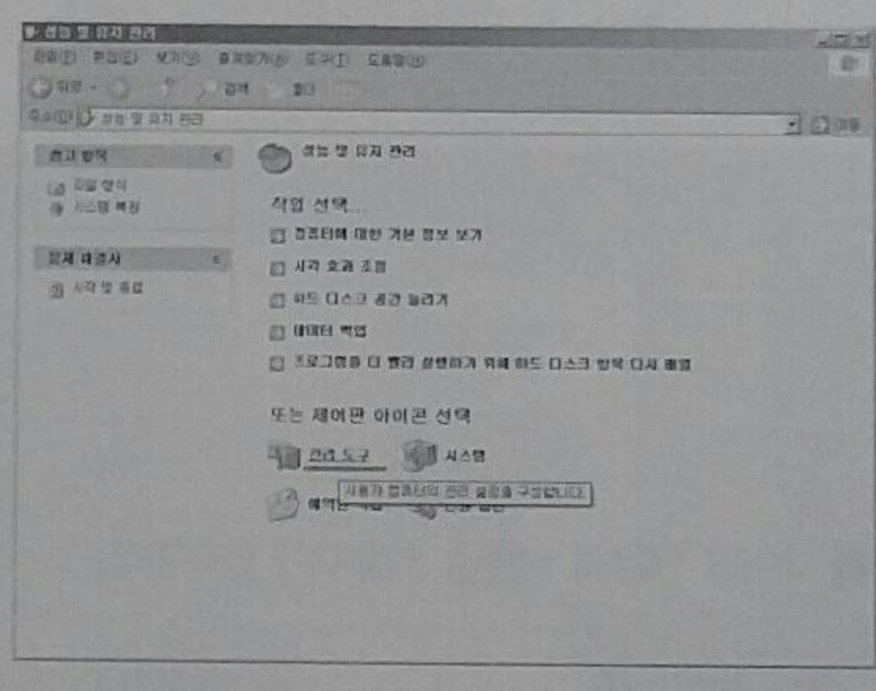
▶ 제어판의 선택

❷ 제어판 메뉴에서 성능 및 유지관리 메뉴를 클릭한다.



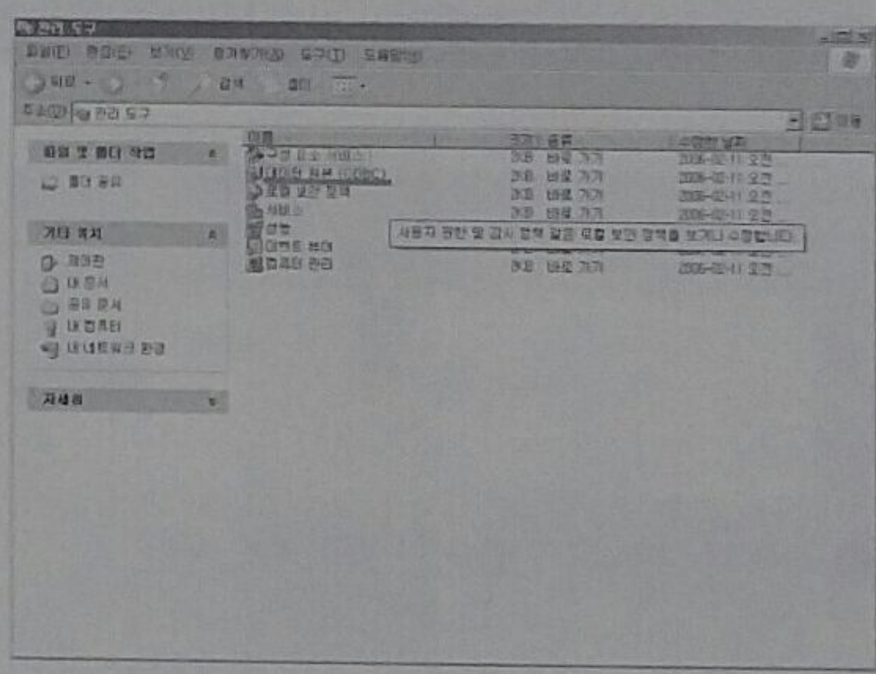
▶ 메뉴선택

❸ 관리도구 메뉴를 선택한다.



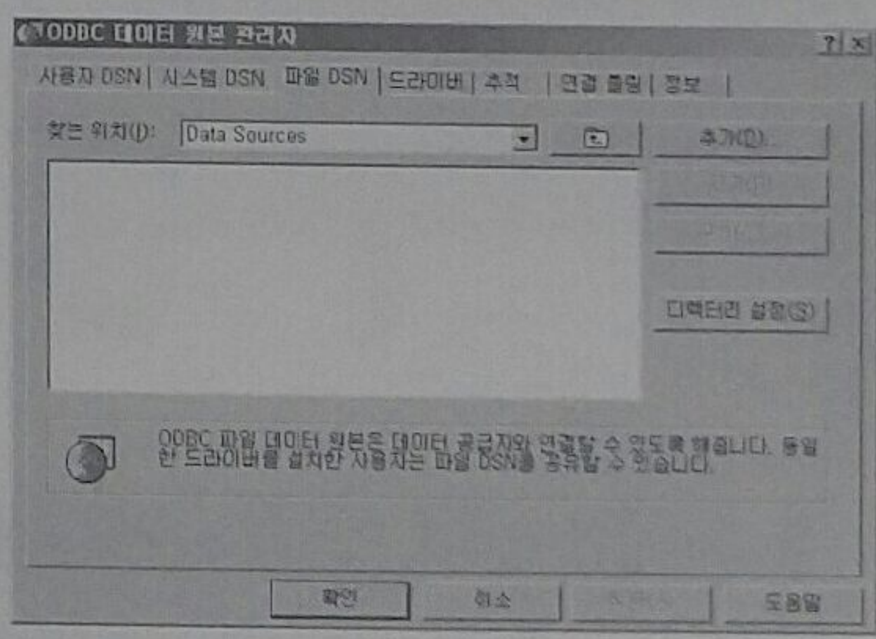
▶ 관리도구 메뉴 선택

❹ 데이터원본정책(ODBC)을 선택한다.



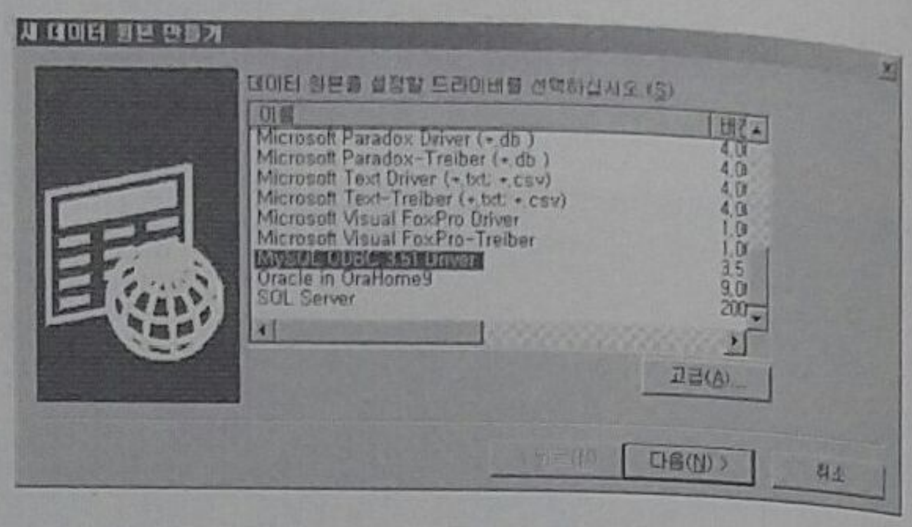
▶ 데이터 원본 메뉴 선택

❺ 사용자 DSN에서 추가 버튼을 클릭한다.



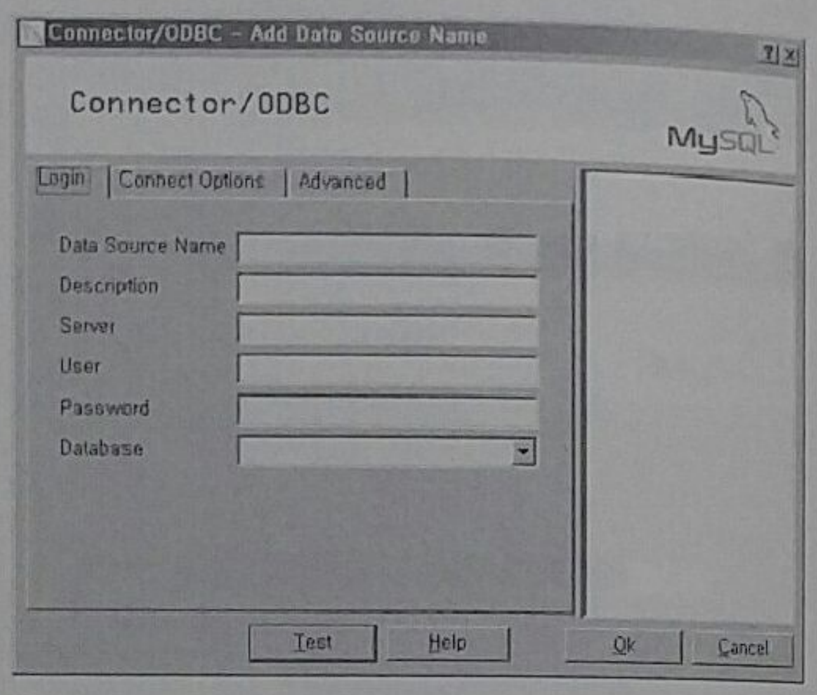
▶ DSN의 추가

❻ 설치된 MySql-ODBC 드라이버를 선택한다.



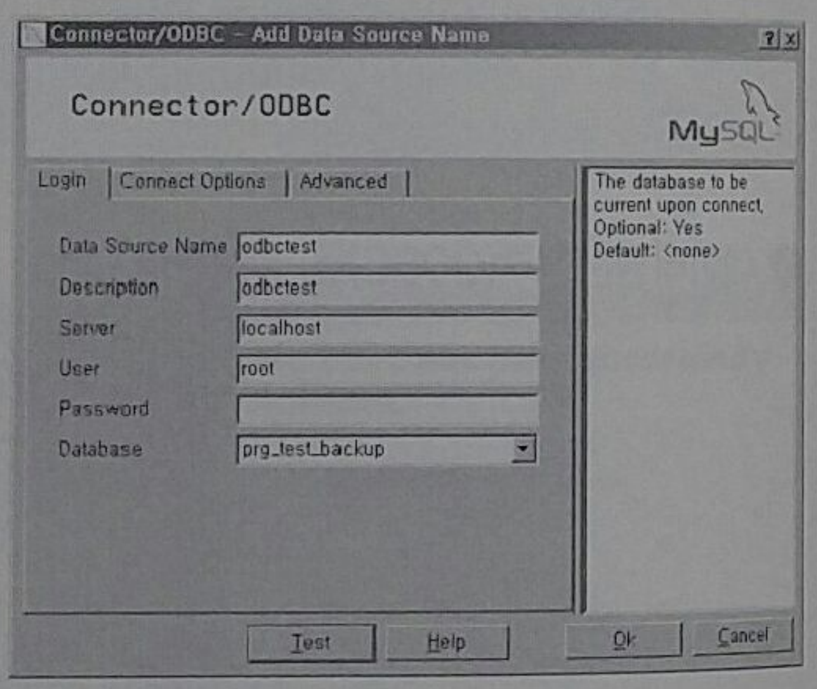
▶ 드라이버의 선택

❼ MySql에 접근하기 위한 Connector 창이 출력이 된다.



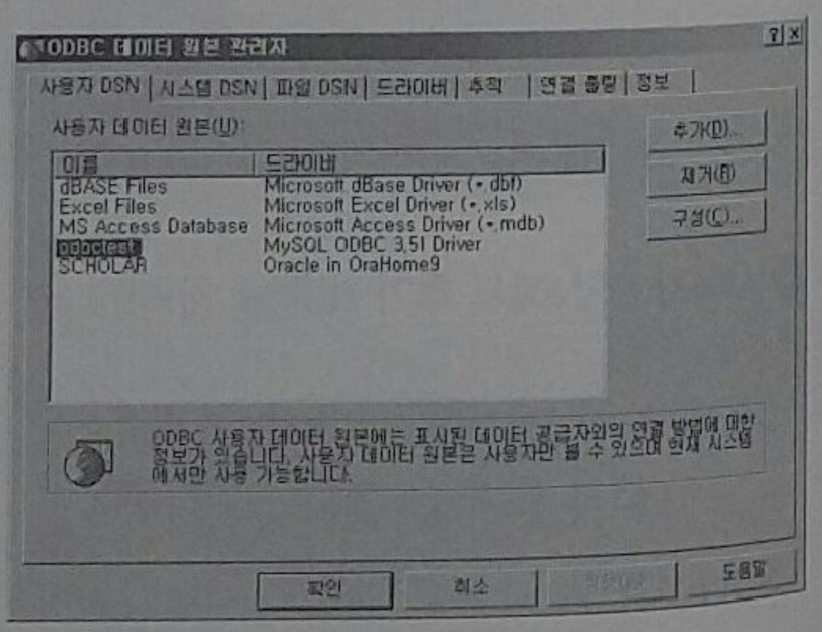
▶ Connector 창

❽ 데이터 원본의 이름을 사용자가 입력한다.



▶ 데이터 원본 이름 입력

❾ 사용자 DSN이 만들어진 것이 확인된다.



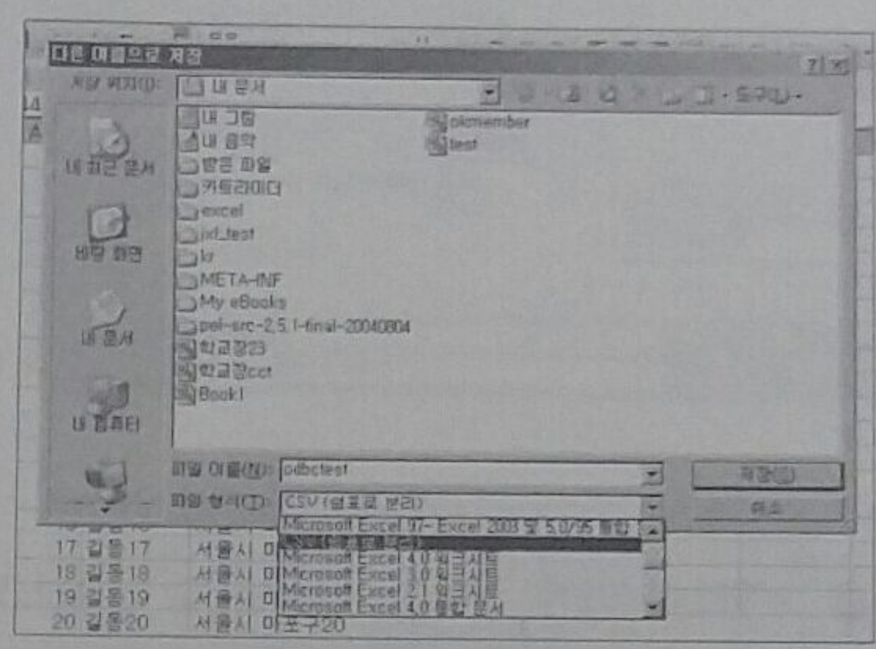
▶ 작성 확인

CSV 데이터 생성과 Access에서 ODBC를 이용한 데이터베이스 연결

엑셀에서 또는 에디터 편집기에서 테이블에 삽입하고자 하는 데이터를 만든 후에 csv 파일로 저장한다.

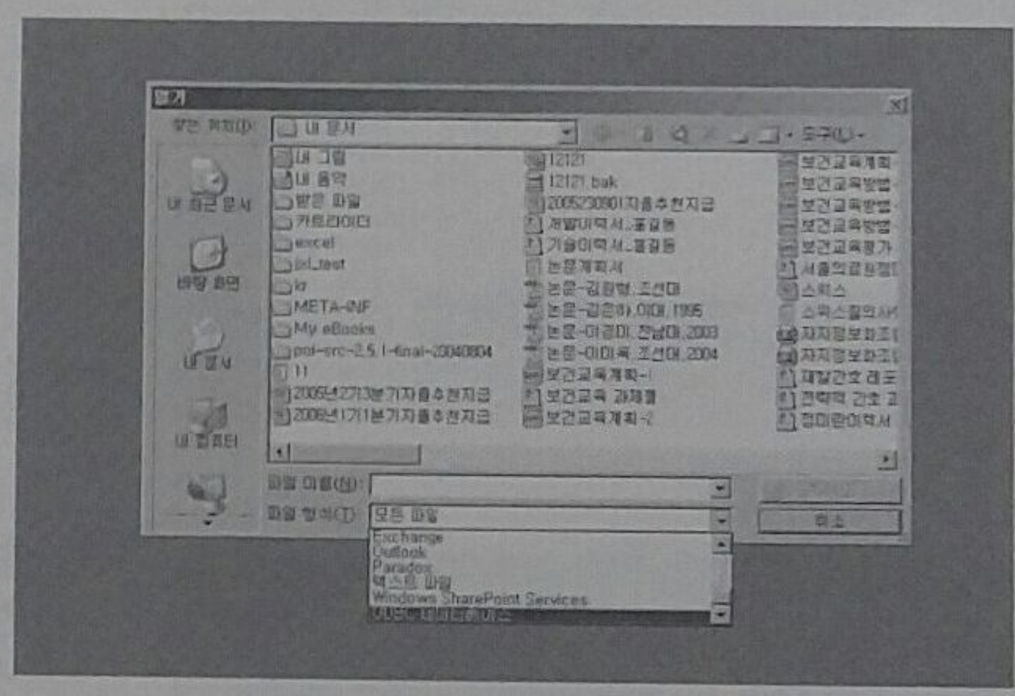
따라하기 액세스에서 ODBC 연결

❶ 편리하게 사용되는 엑셀에서 odbctest 테이블에 들어갈 데이터를 생성한다.



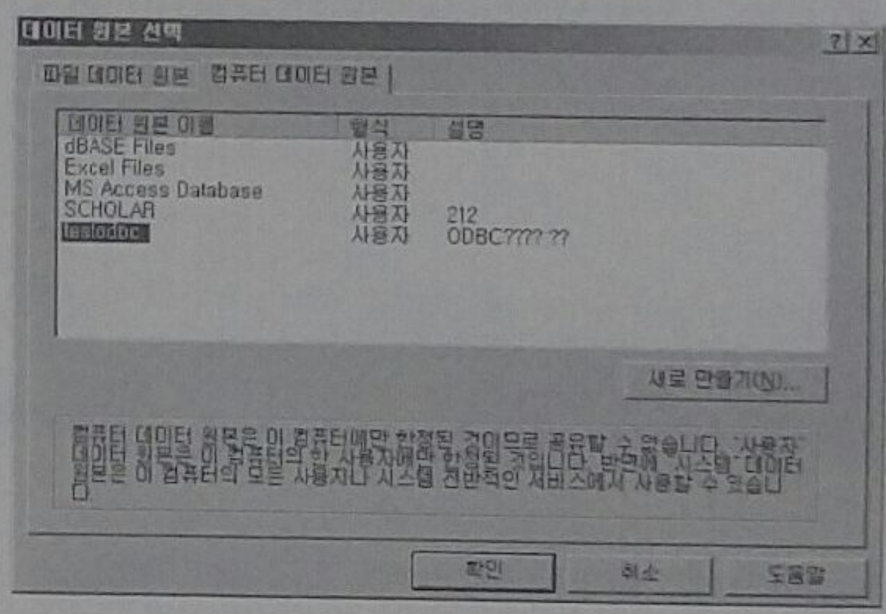
▶ 엑셀에서 데이터 만들기

❷ 만들어진 데이터를 csv 파일로 저장한다.



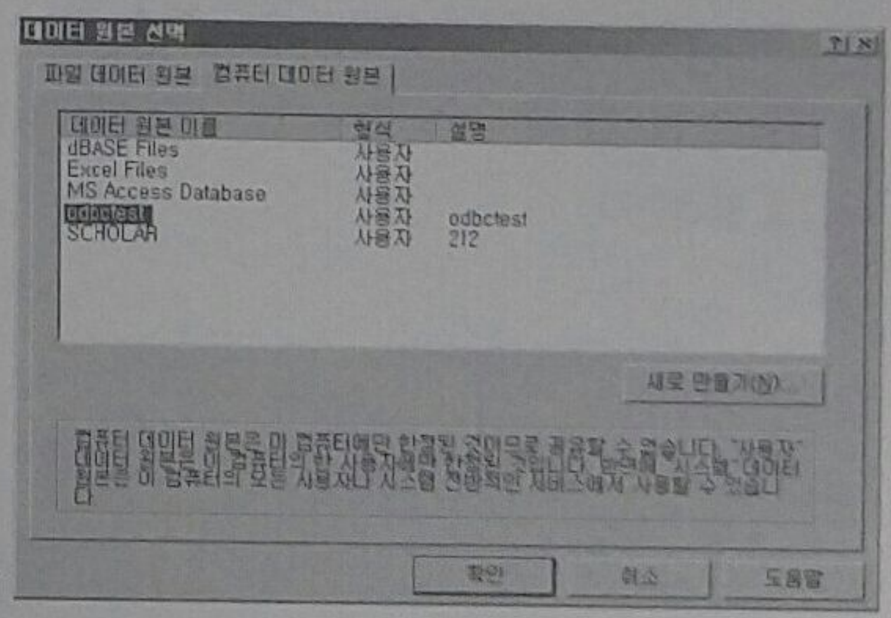
▶ csv 파일 저장

❸ Access를 실행하여 열기->ODBC 데이터베이스를 선택한다.



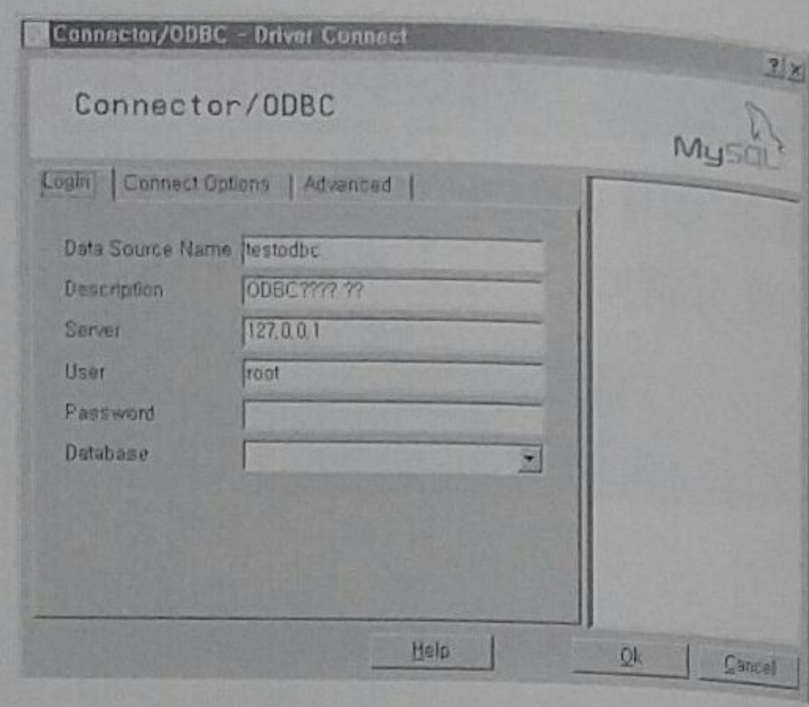
▶ ODBC 데이터베이스 선택

❹ 데이터 원본 창이 나타나면 데이터 원본 선택 창에서 컴퓨터 데이터 원본에서 사용자가 등록한 DSN을 선택한다.



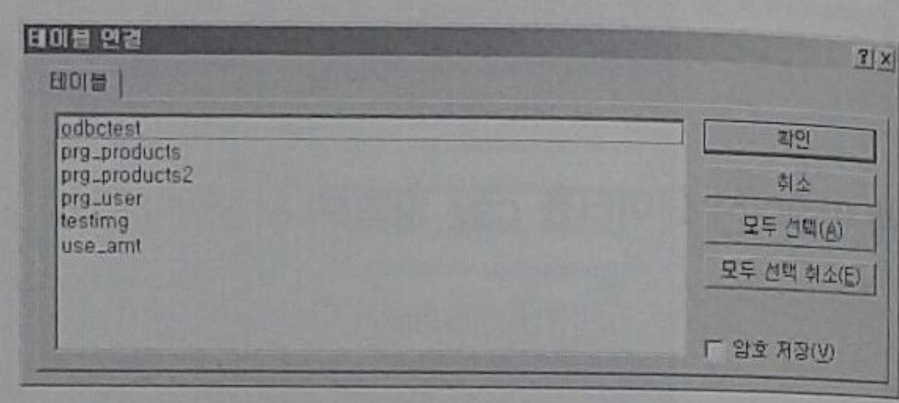
▶ 데이터 원본 창

❺ 테이블 연결 창이 출력된다.



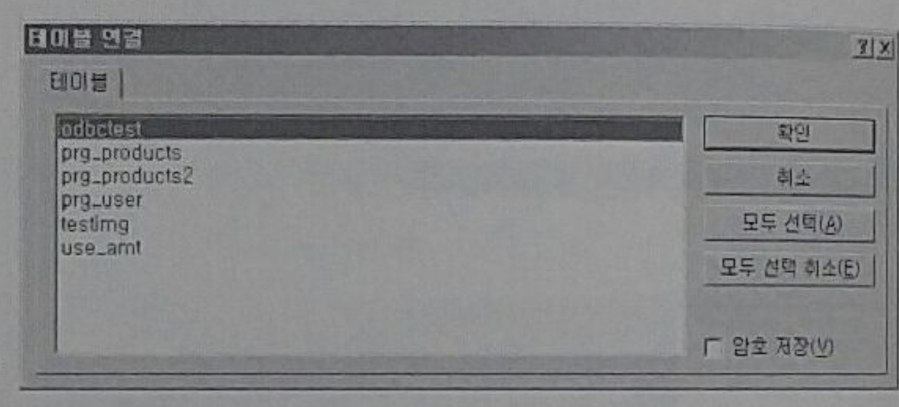
▶ 테이블 연결

❻ 데이터를 삽입할 테이블 odbctest를 선택하고 확인 버튼을 클릭한다.



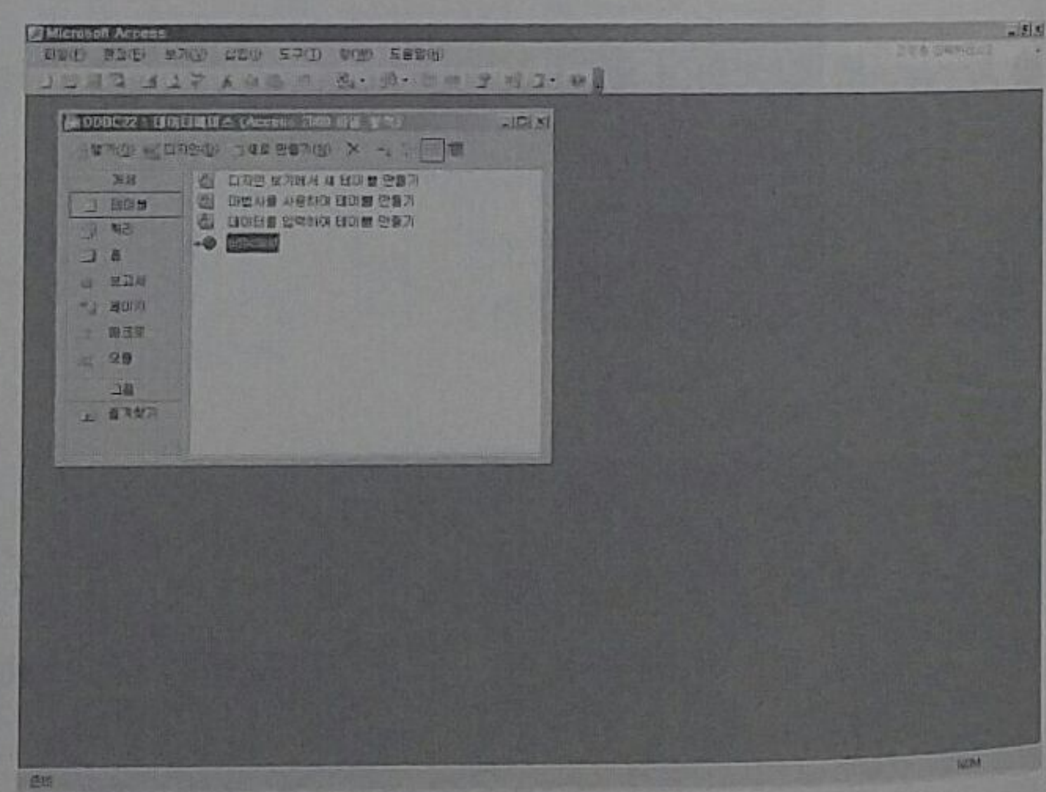
▶ 테이블 선택

❼ 성공적으로 연결된 것을 확인할 수 있다.



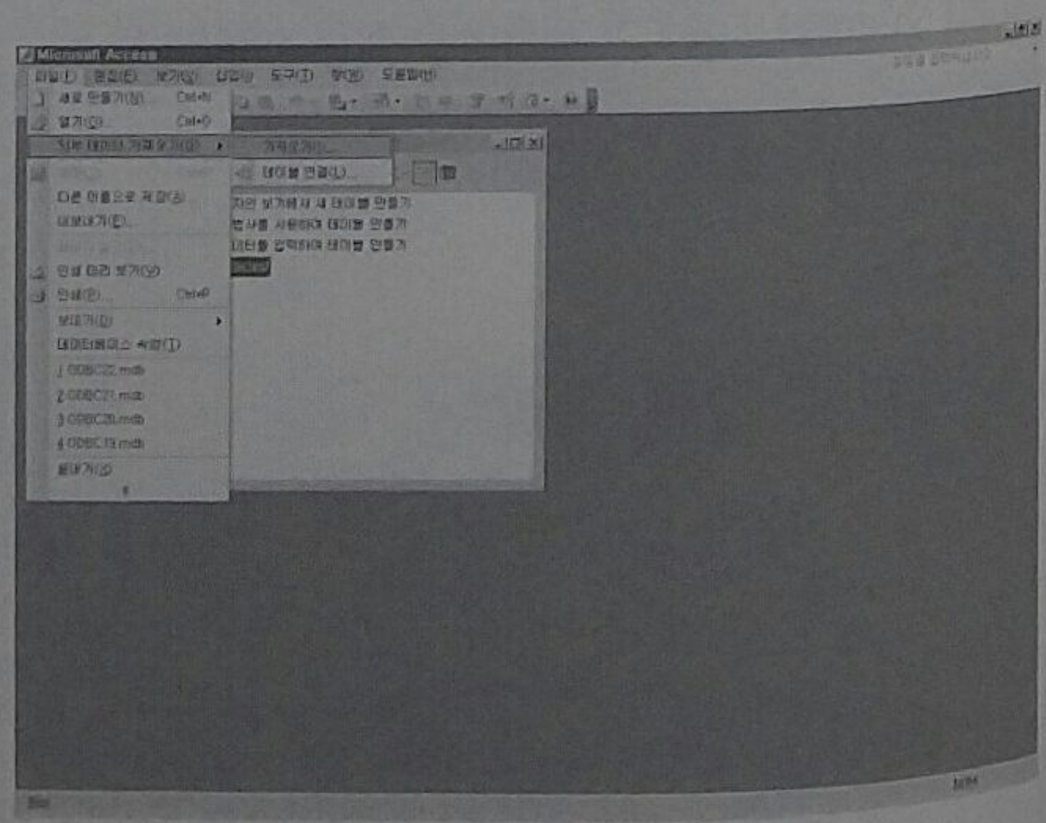
▶ 연결성공

❽ 파일 → 외부 데이터 가져오기 → 가져오기를 선택한다.



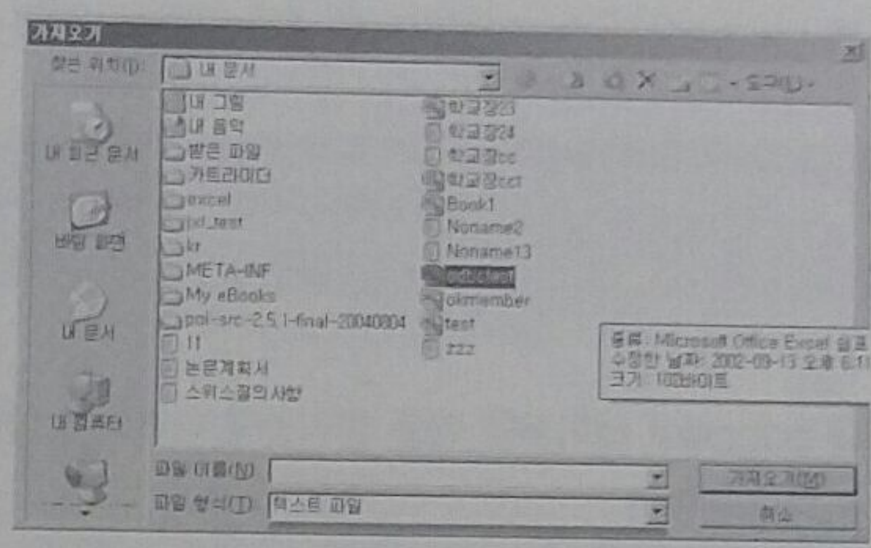
▶ 저장할 파일 가지고 오기

❾ 파일의 형식을 텍스트로 선택하고 csv 파일로 저장된 파일을 불러온다.



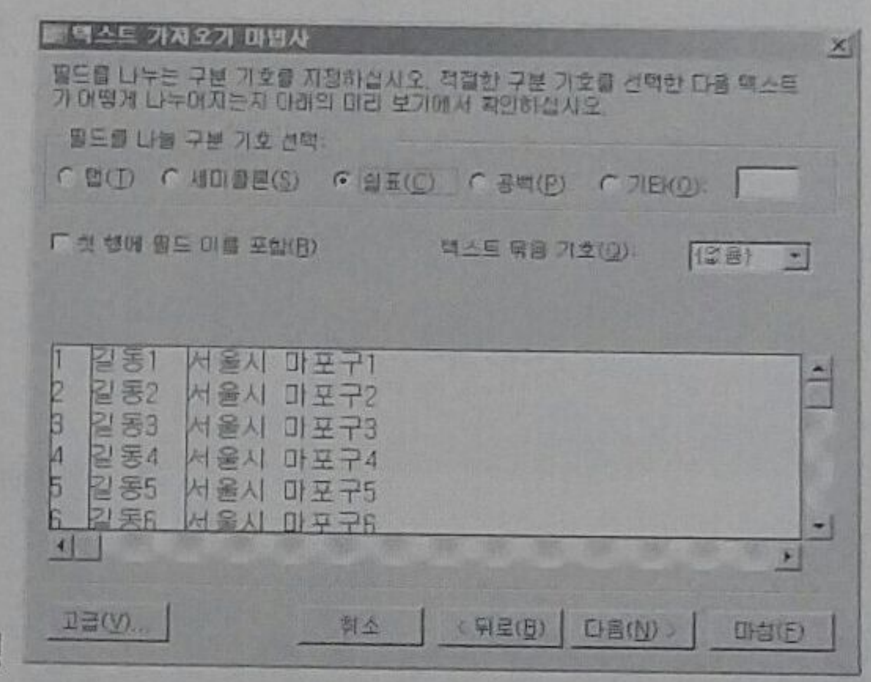
▶ csv 파일의 선택

⑩ 텍스트 가져오기 마법사가 실행이 된다.



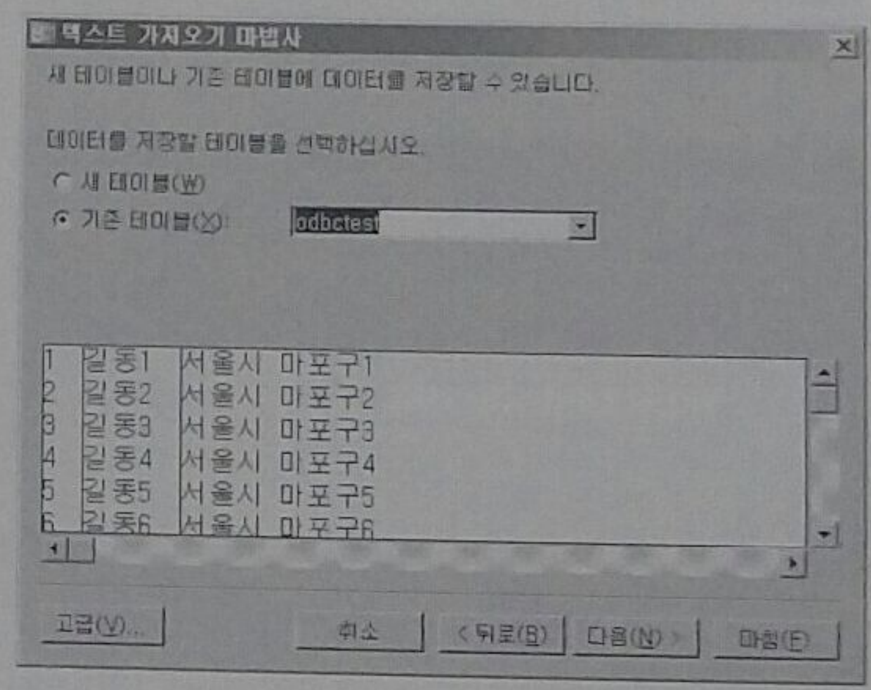
▶ 마법사 실행

⑪ 다음을 클릭하여 구분자를 쉼표로 선택한 후 다음 버튼을 클릭한다.



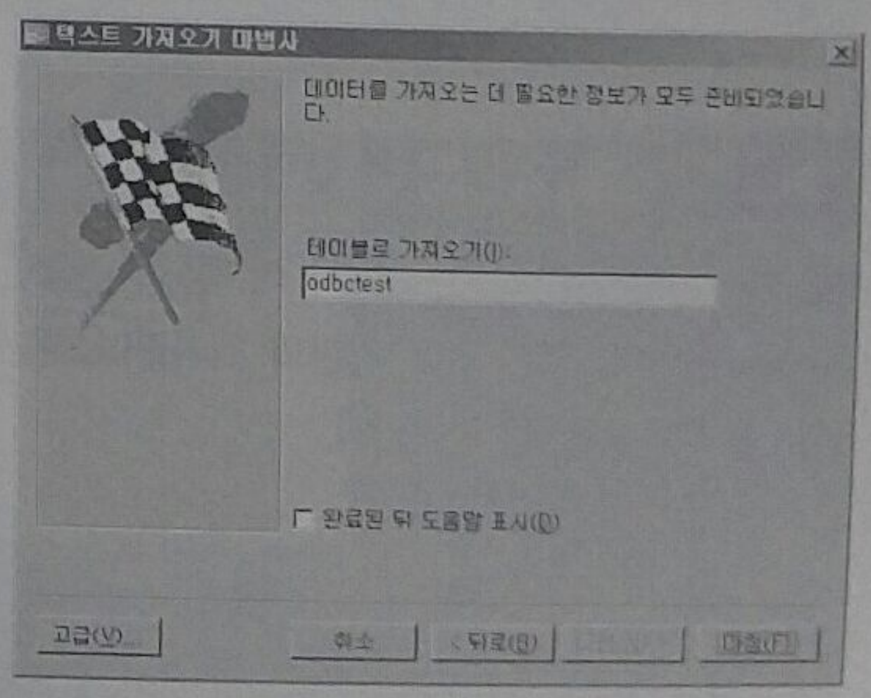
▶ 구분자 지정

⑫ 기존 테이블을 선택하고 데이터를 저장할 테이블을 선택한 후 다음 버튼을 클릭한다.



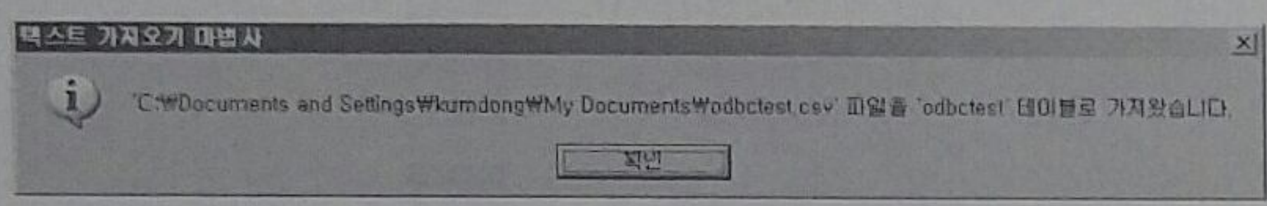
▶ 테이블의 선택

⑬ 테이블 가져오기 마침을 클릭한다.



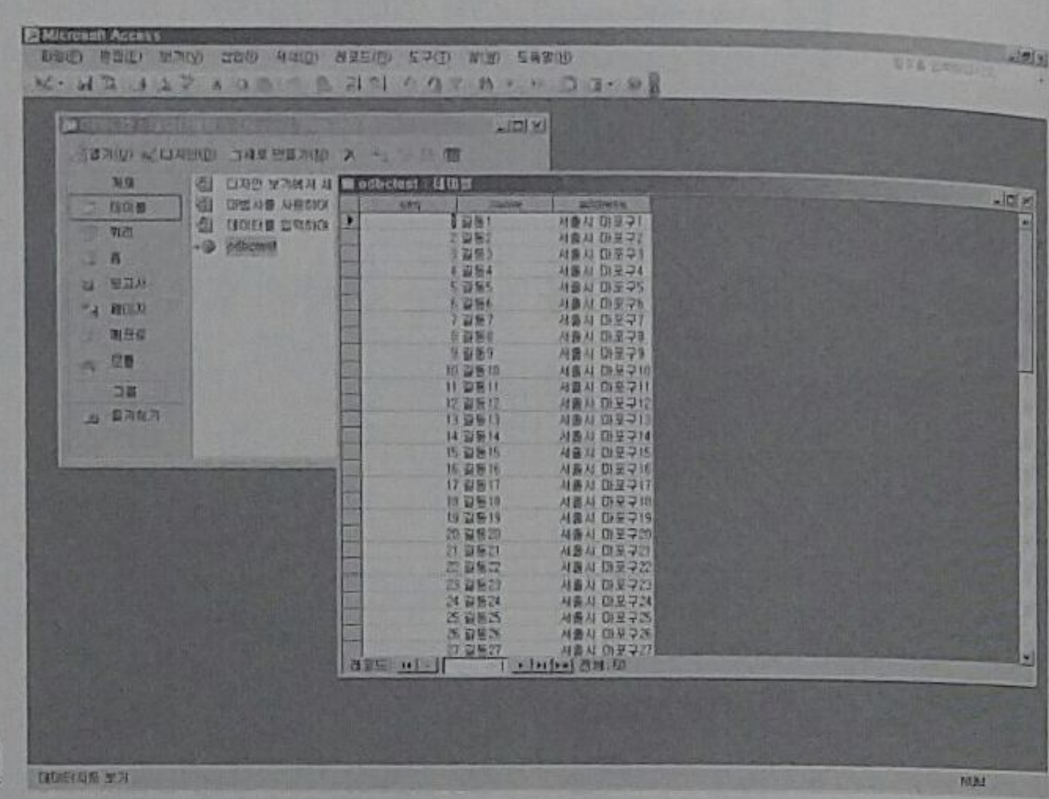
▶ 테이블 가져오기

⑭ 테이블이 성공적으로 가져온 메시지를 출력한다.



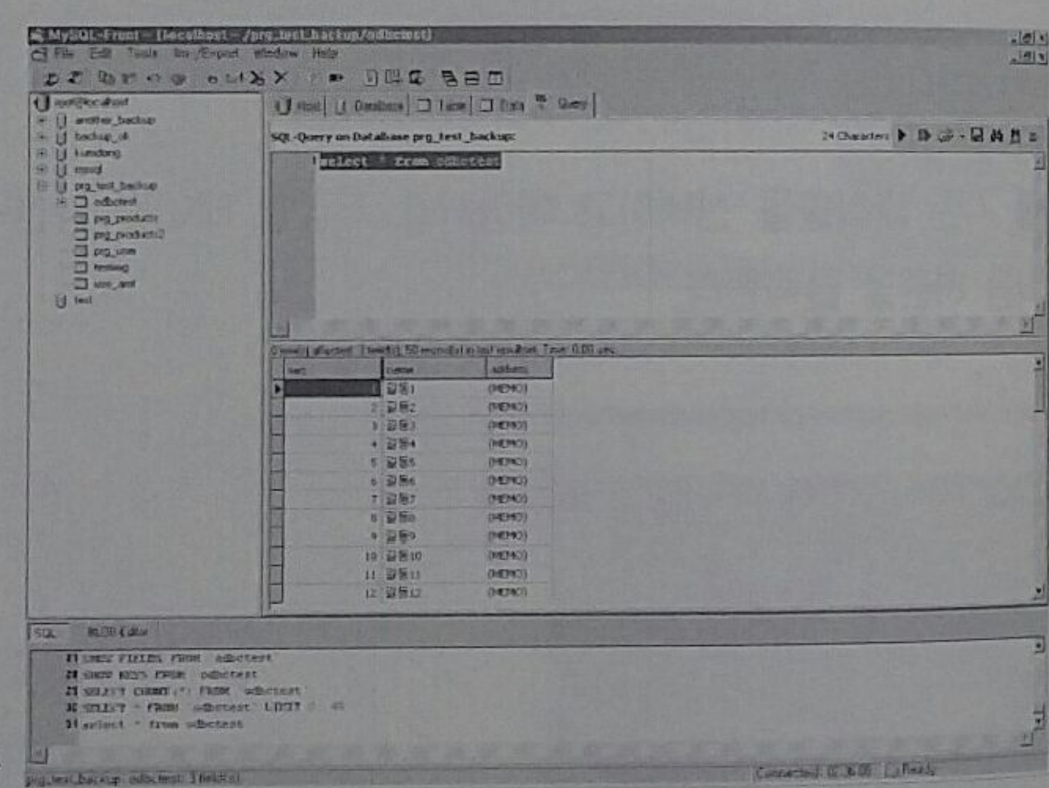
▶ 성공 메시지

⑮ odbctest 테이블을 더블 클릭하여 데이터가 삽입되었는지 확인한다.



▶ 데이터 저장 확인

⑯ Front에 접속하여 select 해 본다.

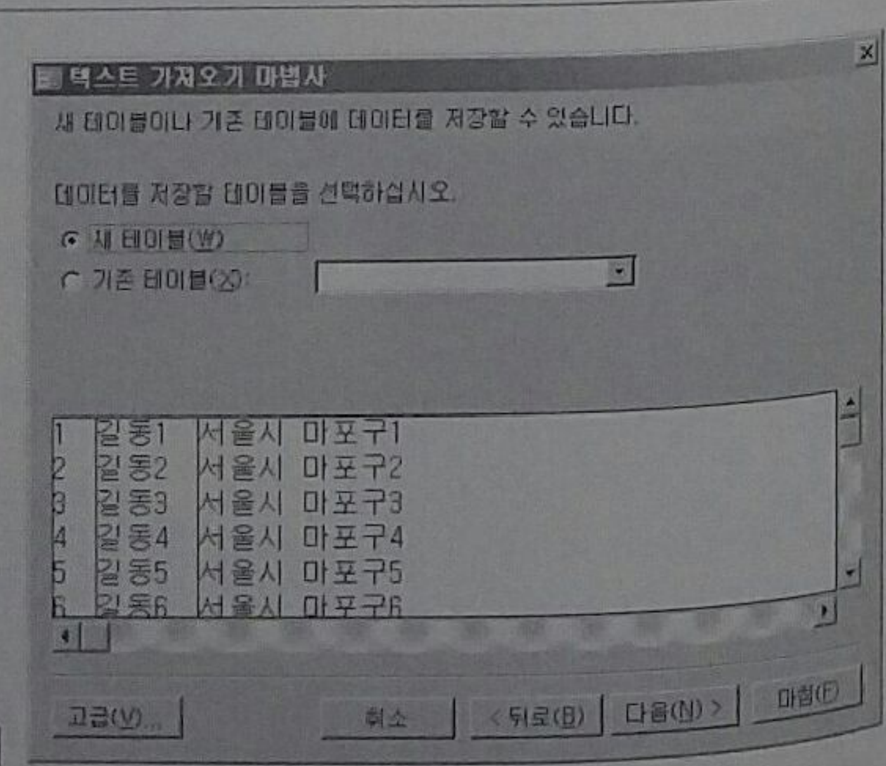


▶ Select 결과

만약에 테이블 내에 데이터가 존재한다면 붙여넣기를 할 수 있다.

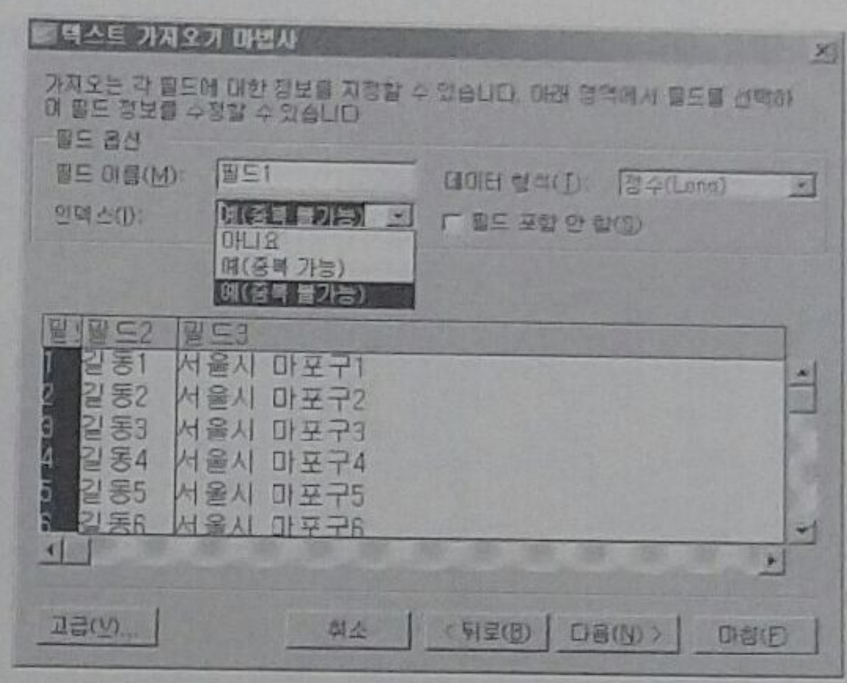
테이블 내에 붙여넣기

❶ 같은 방법으로 Access에서 테이블과 연결한 후 텍스트 파일을 불러들인다. 그런 후에 기존의 테이블이 아닌 새 테이블을 선택한 후 다음 버튼을 클릭한다.



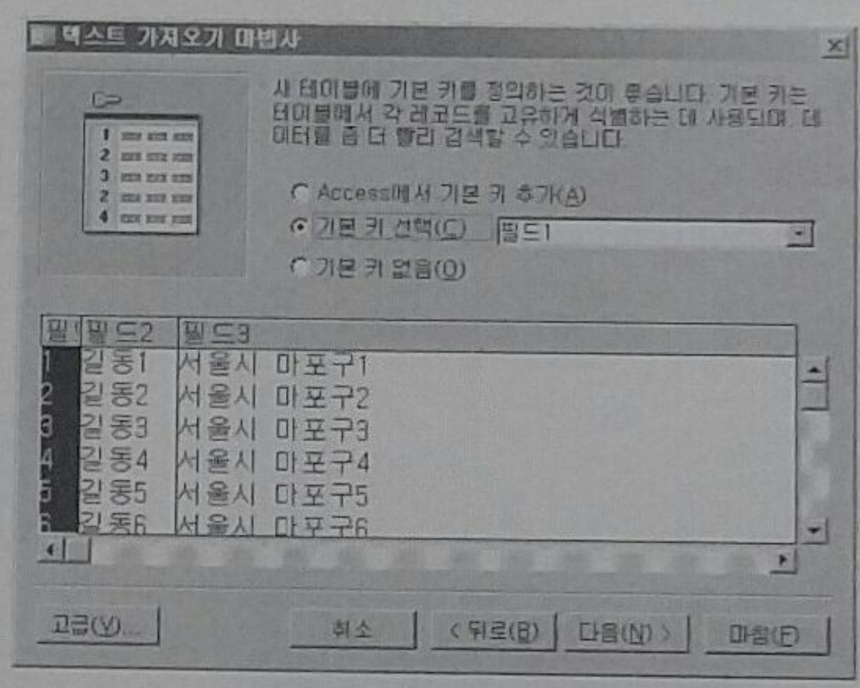
▶ 새 테이블 불러들이기

❷ 필드 옵션을 선택 후 다음 버튼을 클릭한다.



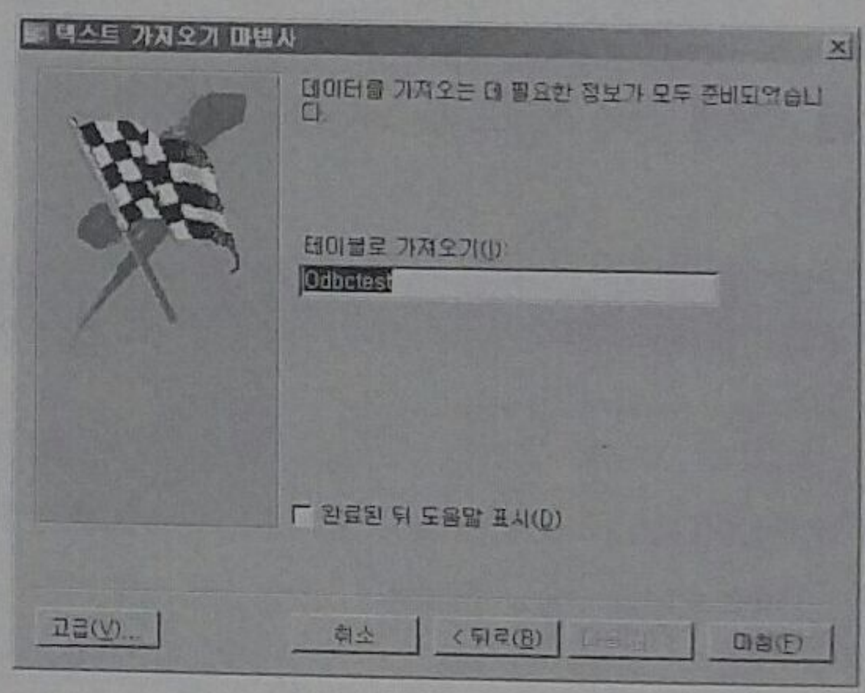
▶ 필드옵션 선택

❸ 기본키를 지정한다.



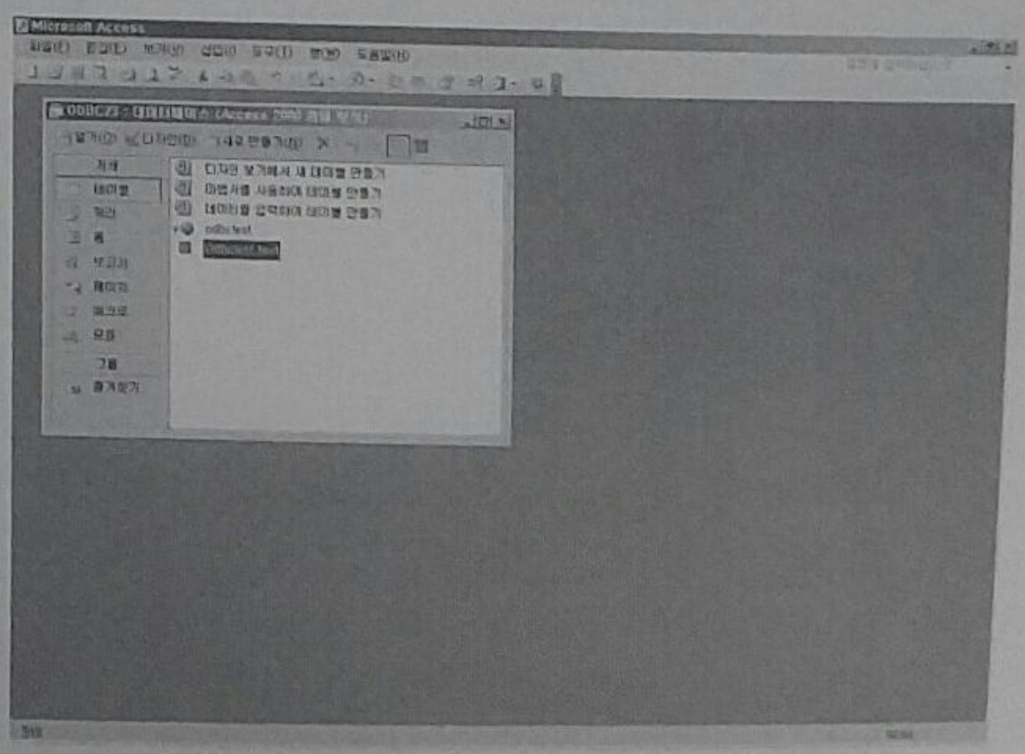
▶ 기본키 지정

❹ 데이터 가져오기를 한다. 데이터를 가지고 올 때 이름을 다르게 주어야 한다.



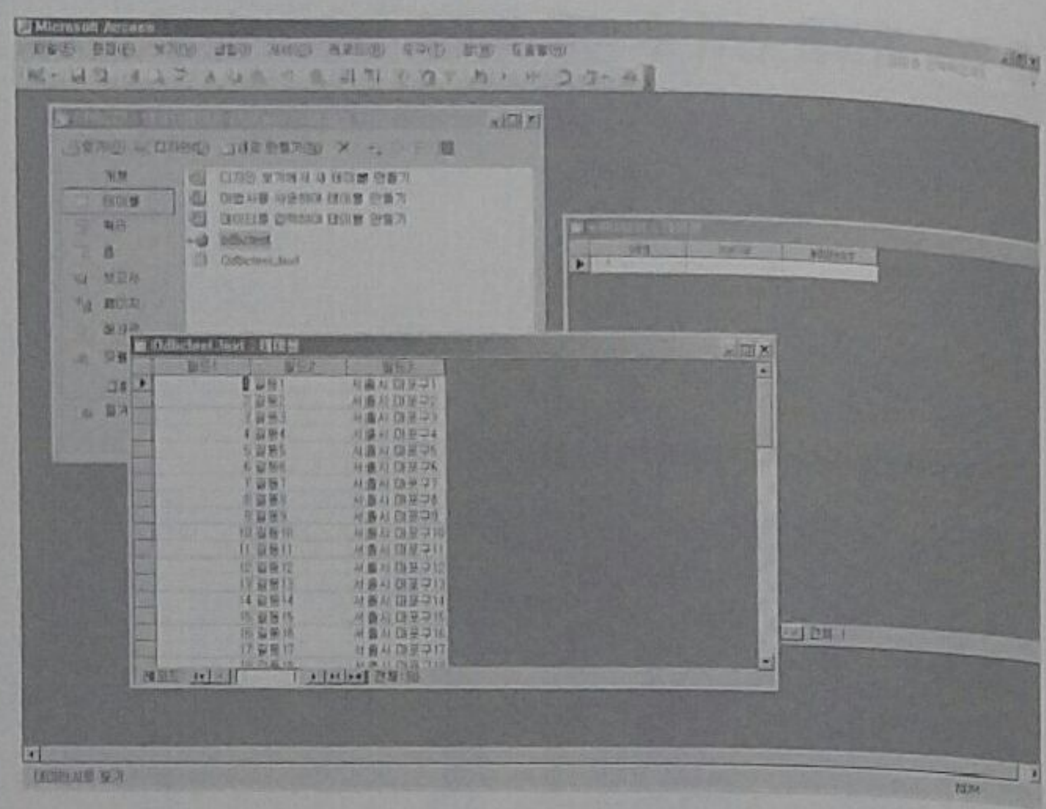
▶ 데이터 가져오기

❺ 데이터 가져오기를 확인한다.



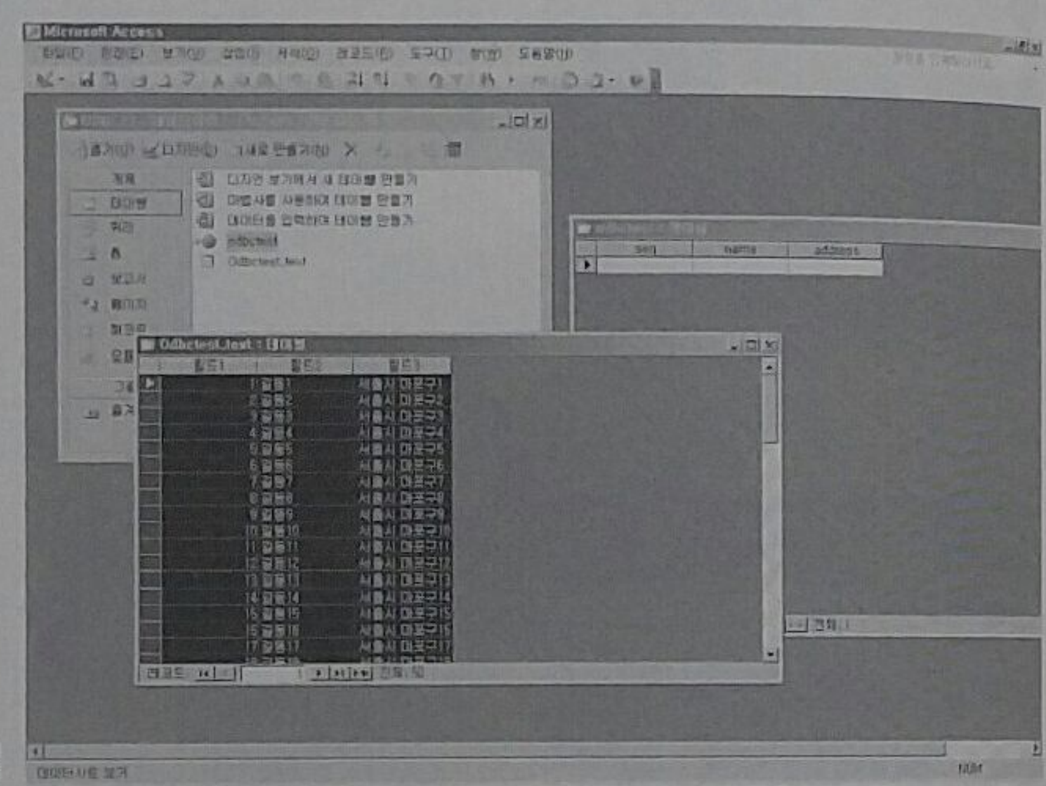
▶ 데이터 가져오기 확인

❻ 열기 또는 더블 클릭하여 두개의 테이블을 연다.



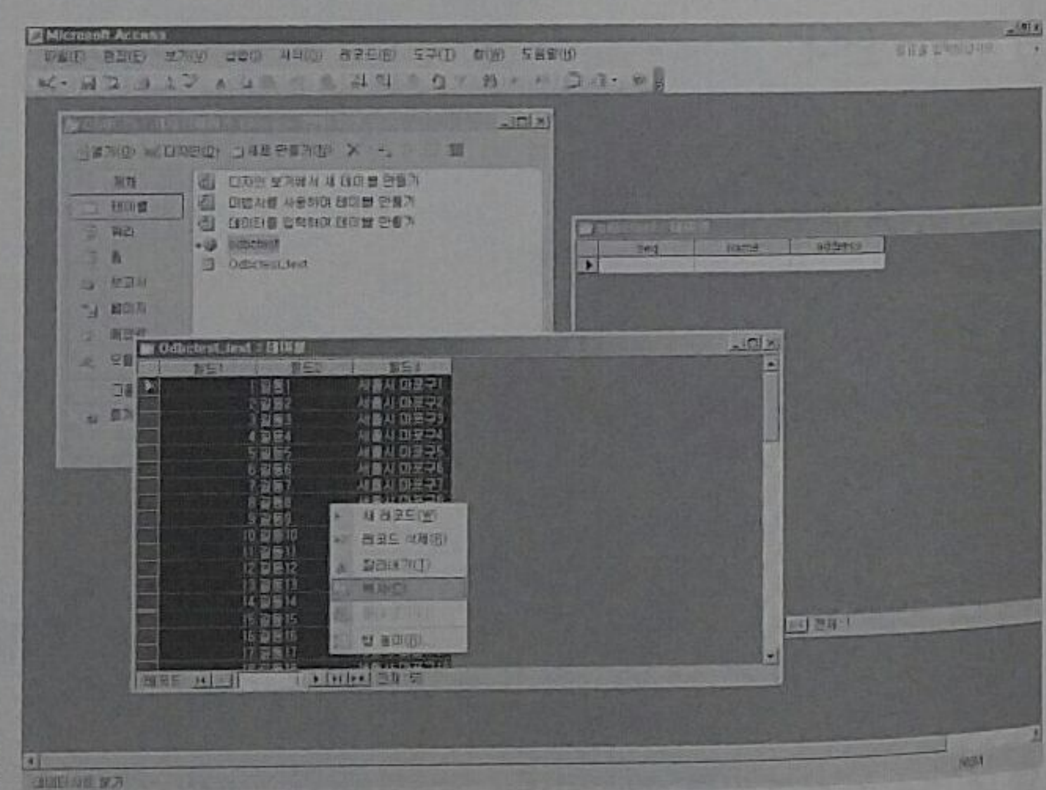
▶ 테이블 열기

❼ 불러들인 텍스트 테이블에서 Ctrl+A를 눌러서 전체 레코드를 선택한다.



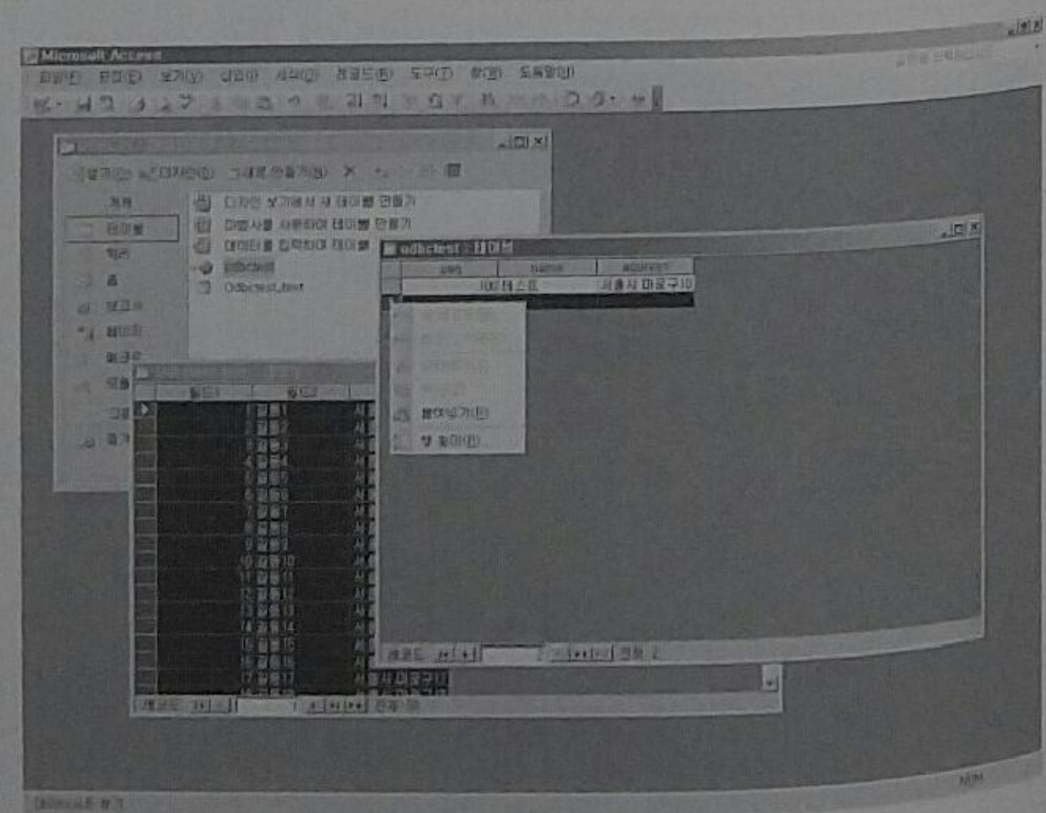
▶ 전체 레코드 선택

❽ 선택한 전체 레코드를 복사한다.



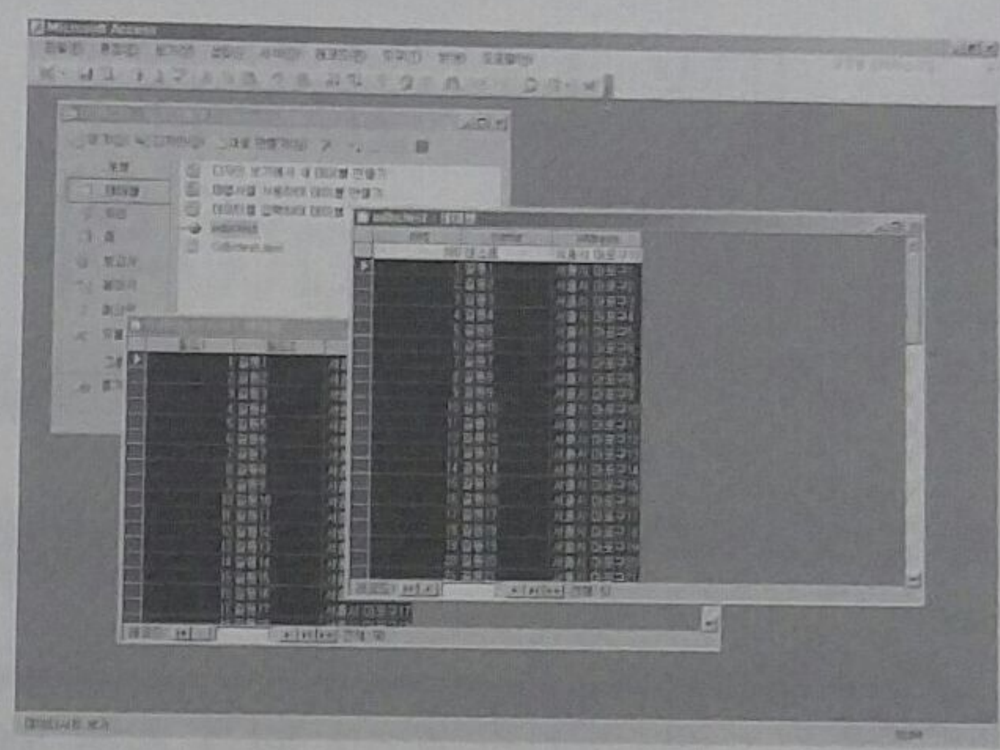
▶ 복사

❾ 삽입할 테이블을 선택하고 붙여 넣기를 한다.



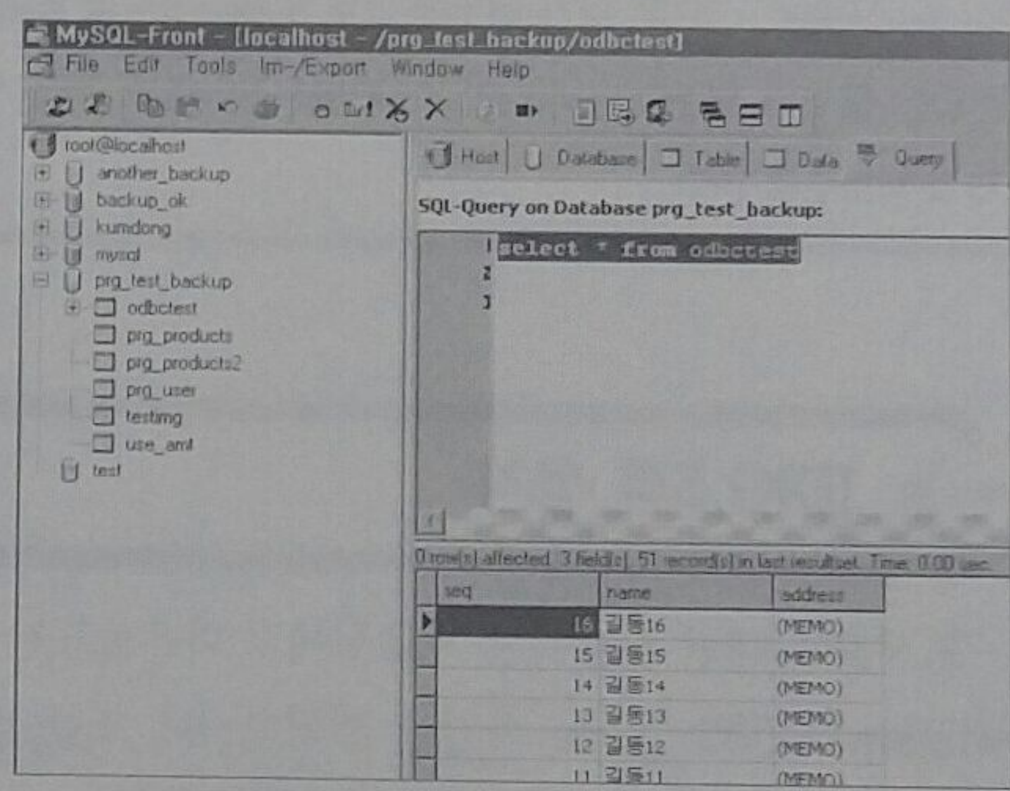
▶ 붙여넣기

⑩ 데이터 붙여넣기가 완료되었다.



▶ 붙여넣기 성공

⑪ Front에서 select한다.



▶ select 결과

맺음말

이번 장에서는 Front의 기타 기능과 ODBC를 통해 MySql 접속하여 데이터를 일괄적으로 삽입하는 방법에 대해서 공부하였다. 데이터를 일괄적으로 삽입하는 방법은 엑셀의 데이터 또는 csv 파일로 생성된 데이터를 효율적으로 빠르게 데이터의 테이블에 삽입하는 방법이므로 이를 알고 있으면 유용하게 사용할 수 있다. 이제 MySql에 대한 기본적인 모든 부분에 대해서 거의 공부했다. 7장부터는 이런 MySql을 이용한 프로그래밍을 위한 환경 설정부터 소개한다.

07 JDK 1.5 설치와 Tomcat5의 설치

안녕하세요

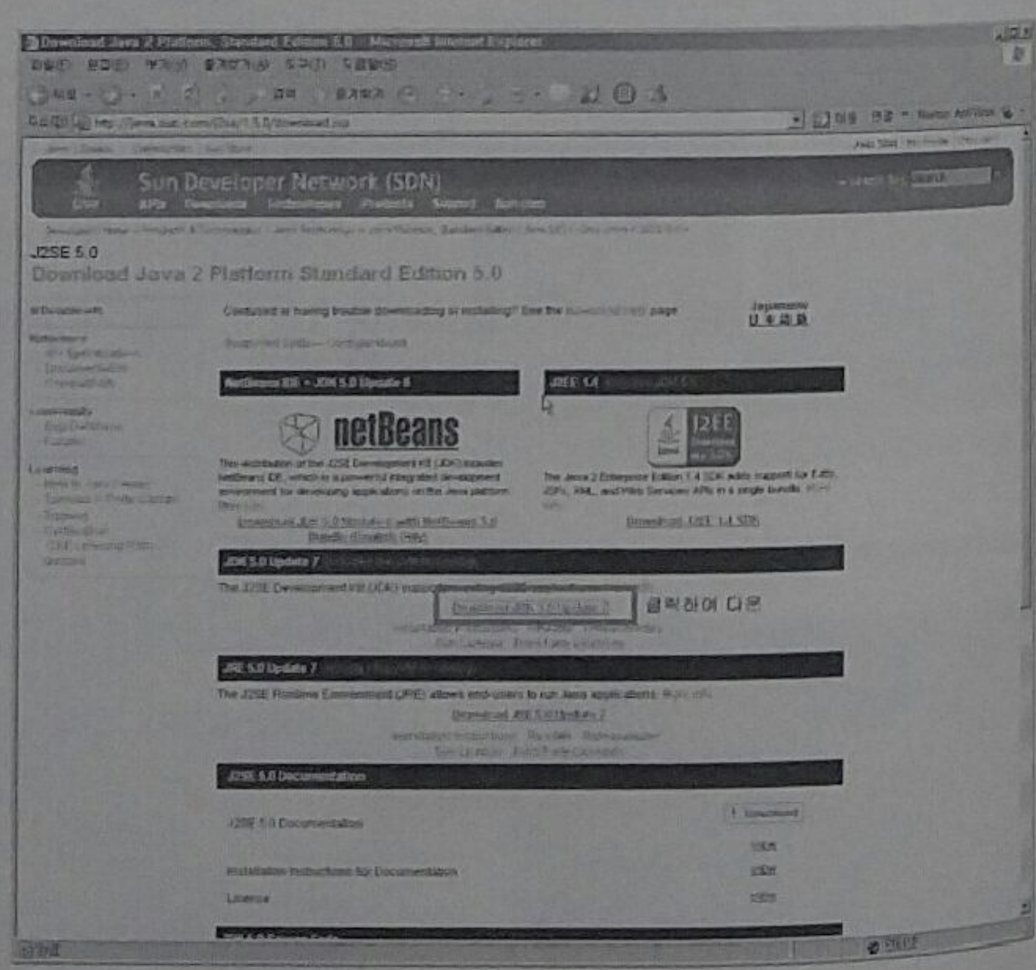
이번 장에서는 웹 프로그램 중 JSP를 개발하기 위하여 필요한 S/W를 설치한다. JSP는 JAVA를 기반으로 하기 때문에 JDK가 필수적으로 설치가 되어야 하며 웹서비스를 위하여 인터넷에서 무상으로 제공되는 웹서버 톰켓의 설치방법에 대해서 공부하게 된다. 앞에서 이야기했지만 공부를 하기 위한 세팅의 절차는 반드시 숙지하도록 한다.

JDK1.5의 설치

도서출판 대림 홈페이지 자료실에서 다운로드한 유틸리티 중 jdk1.5 폴더 안에 있는 jdk-1_5_0_07-windows-i586-p.exe 파일을 실행하거나 또는 http://java.sun.com/j2se/1.5.0/download.jsp를 방문하여 다운로드를 받아 저장한 후에 실행한다.

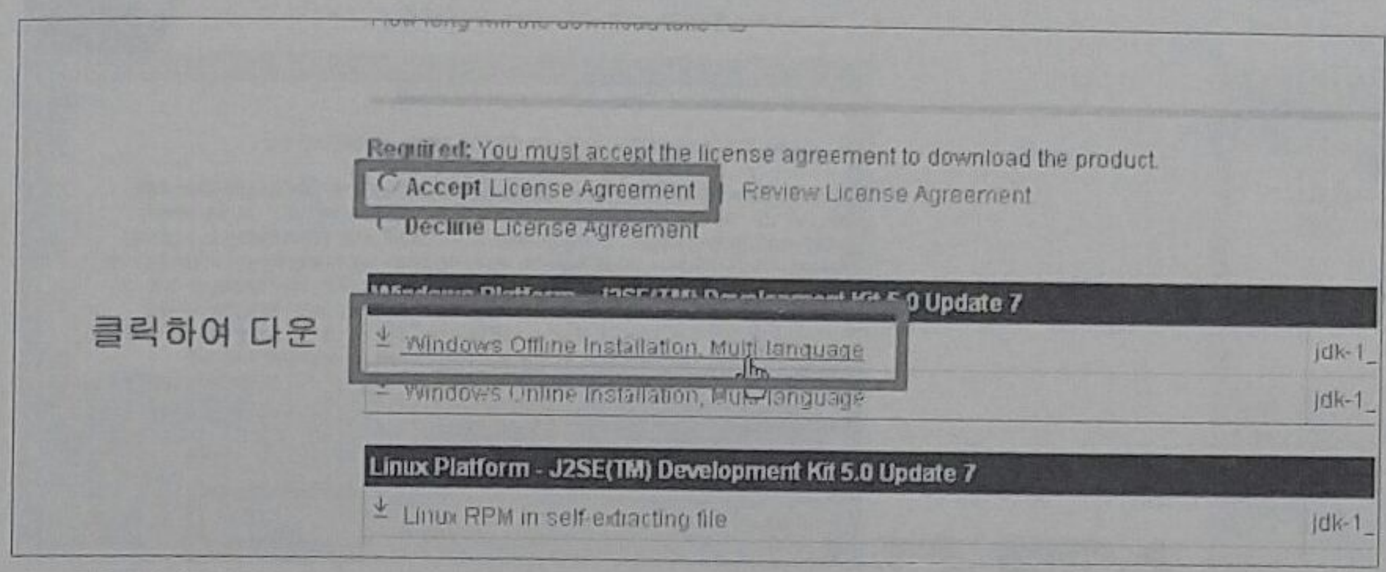
따라하기 JDK 설치

❶ http://java.sun.com/j2se/1.5.0/download.jsp에 접속하여 jdk5.0을 클릭한다.



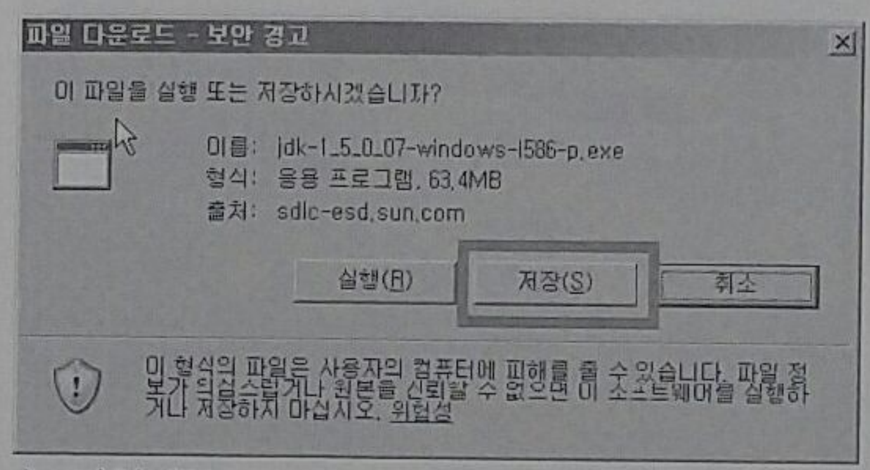
▶ http://java.sun.com/j2se/1.5.0/download.jsp 접속

❷ Accept를 선택한 후 윈도우용 JDK를 클릭한다.



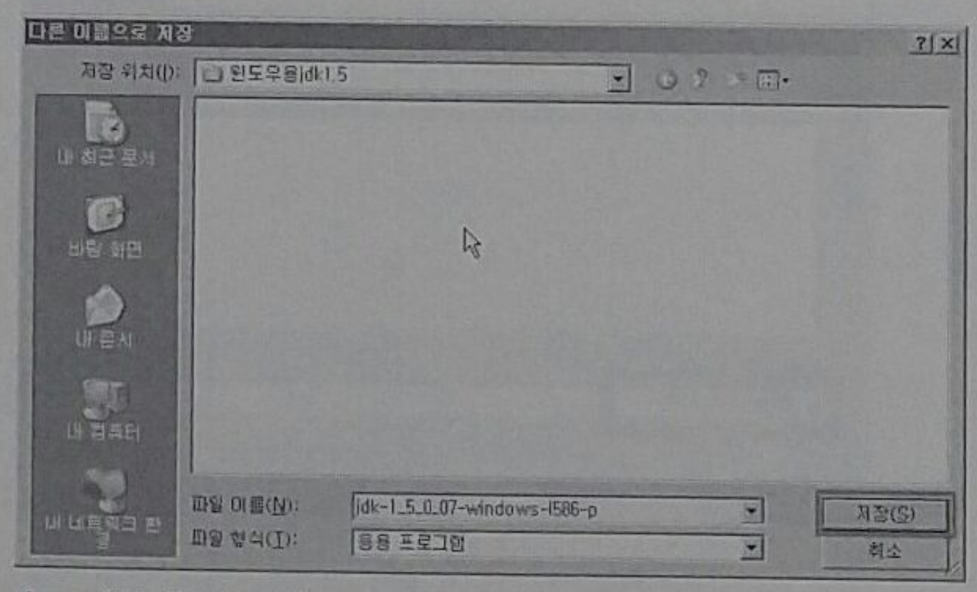
▶ 윈도우용 jdk1.5

❸ 다운받을 파일을 저장한다.

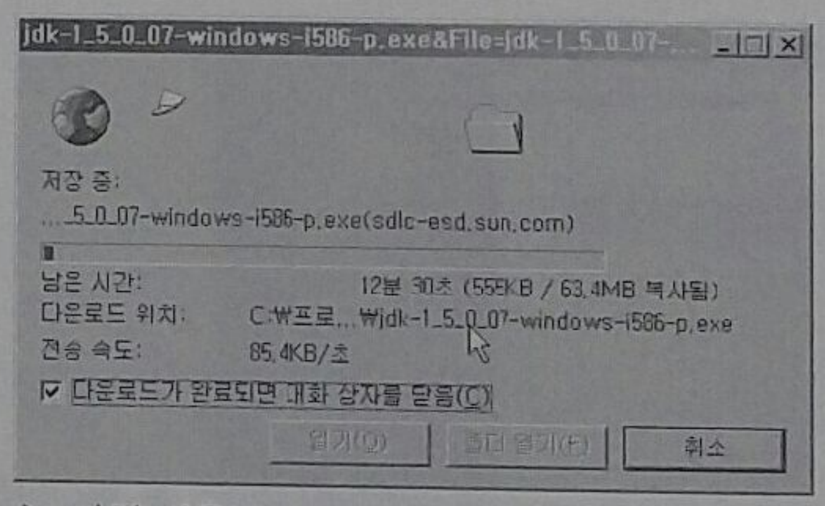


▶ 파일저장

❹ 다운받을 파일을 사용자가 만든 또는 만들어진 폴더를 지정하고 저장 버튼을 클릭한다.

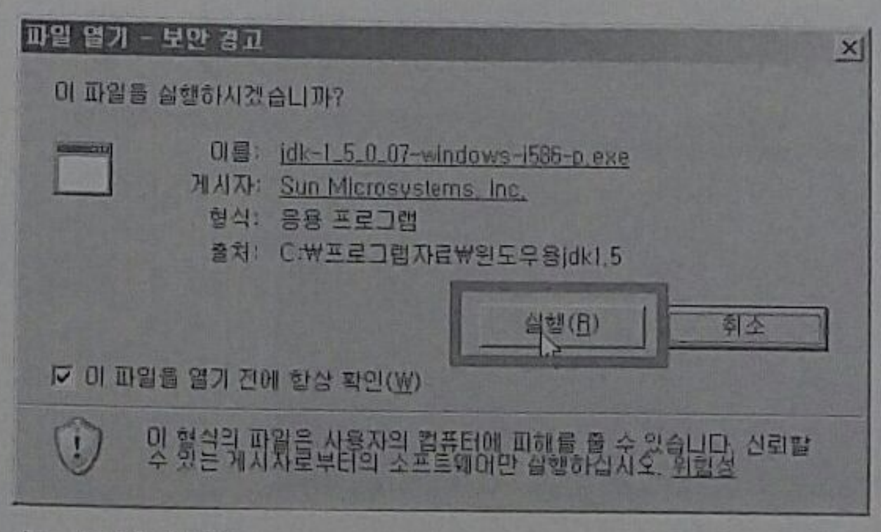


▶ 저장 폴더 지정



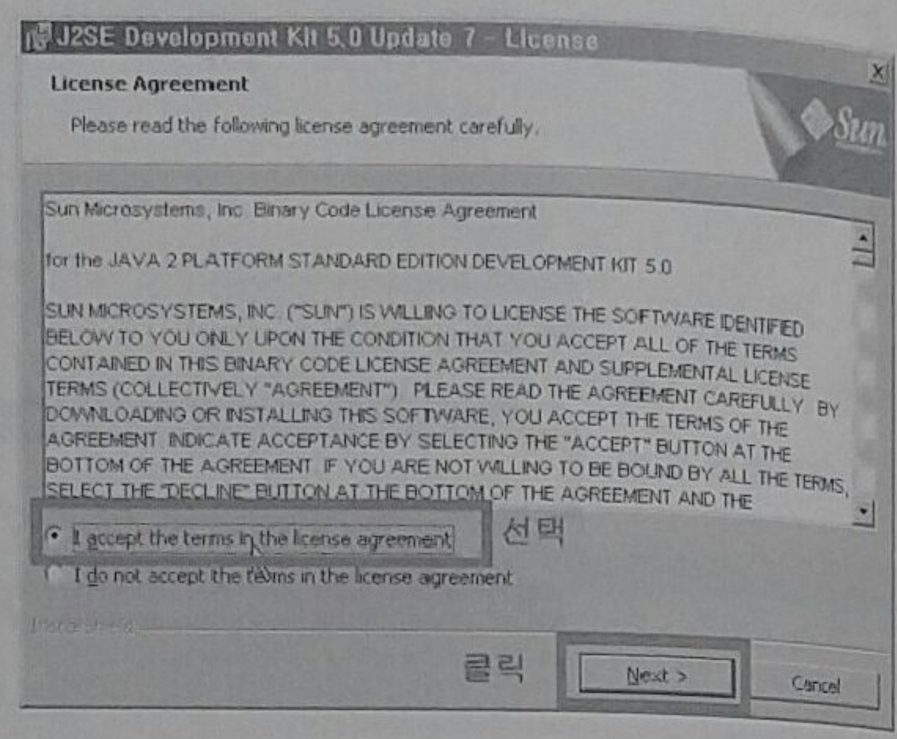
▶ 파일 다운로드

❺ 다운받은 파일을 클릭하여 실행버튼을 클릭한다.



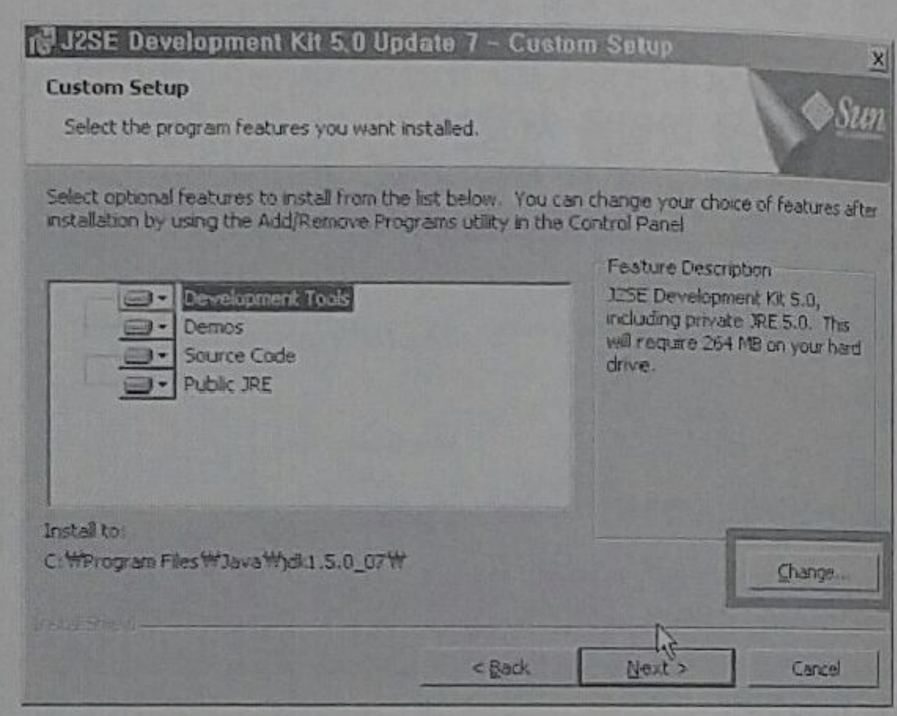
▶ 설치 실행

❻ 라이센스에 동의하고 next 버튼을 클릭한다.



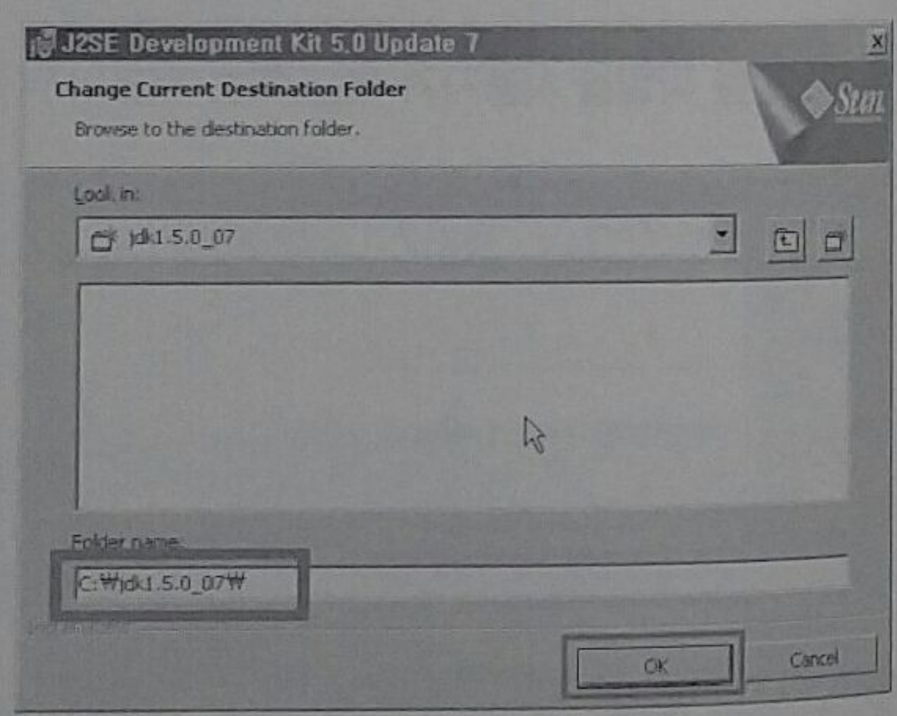
▶ 라이센스 동의

❼ 설치 폴더를 바꾸기 위해 change 버튼을 클릭한다.



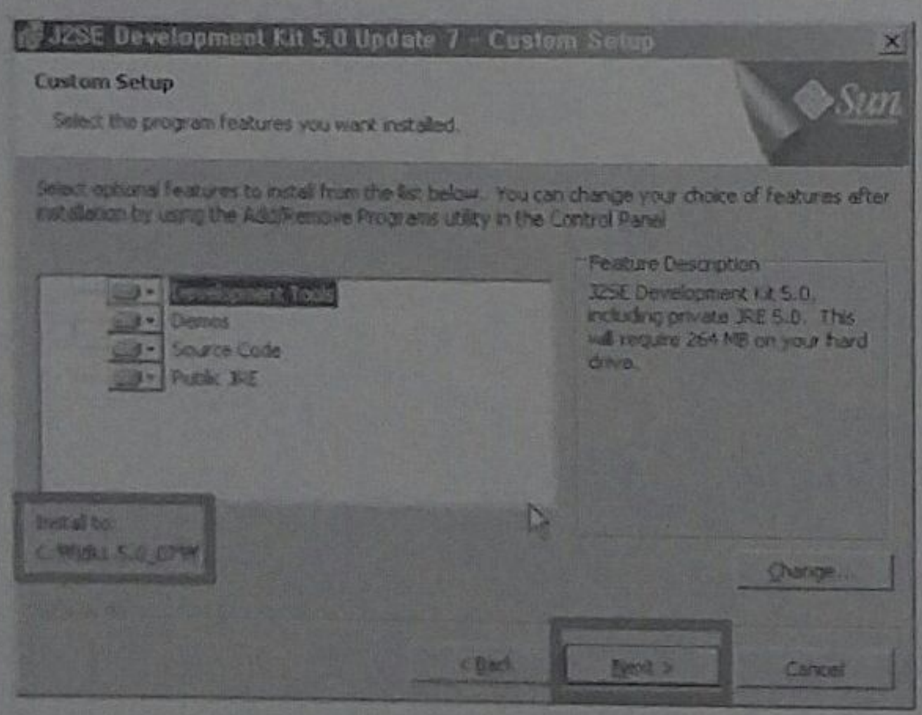
▶ 폴더의 변경

❽ 설치 폴더의 이름을 변경하고 ok 버튼을 클릭한다.

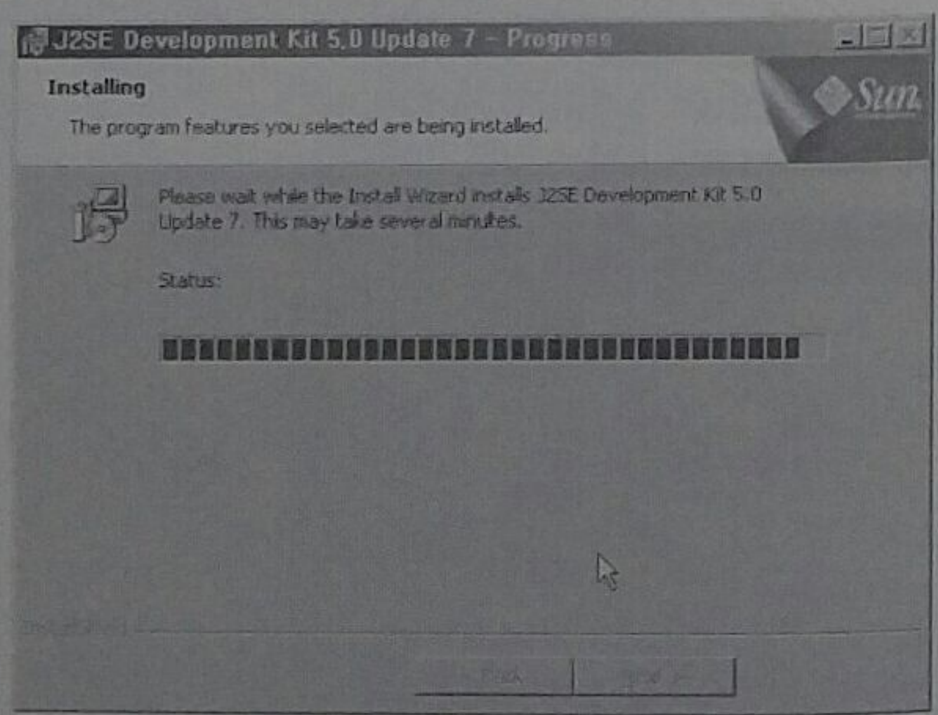


▶ 폴더이름 변경

❾ 설치 폴더가 변경된 것을 확인하고 next 버튼을 클릭한다.

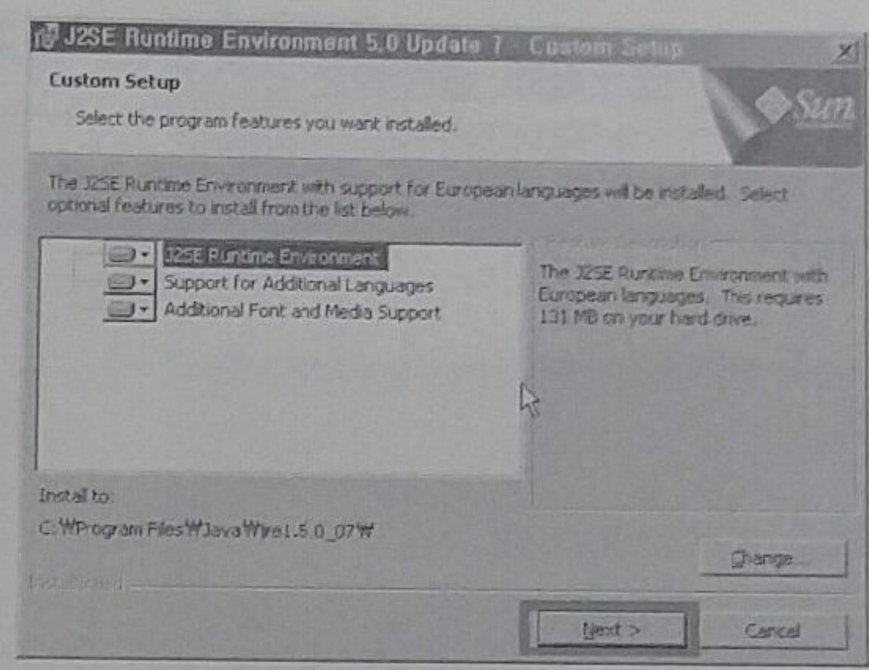


▶ 설치 폴더 변경확인



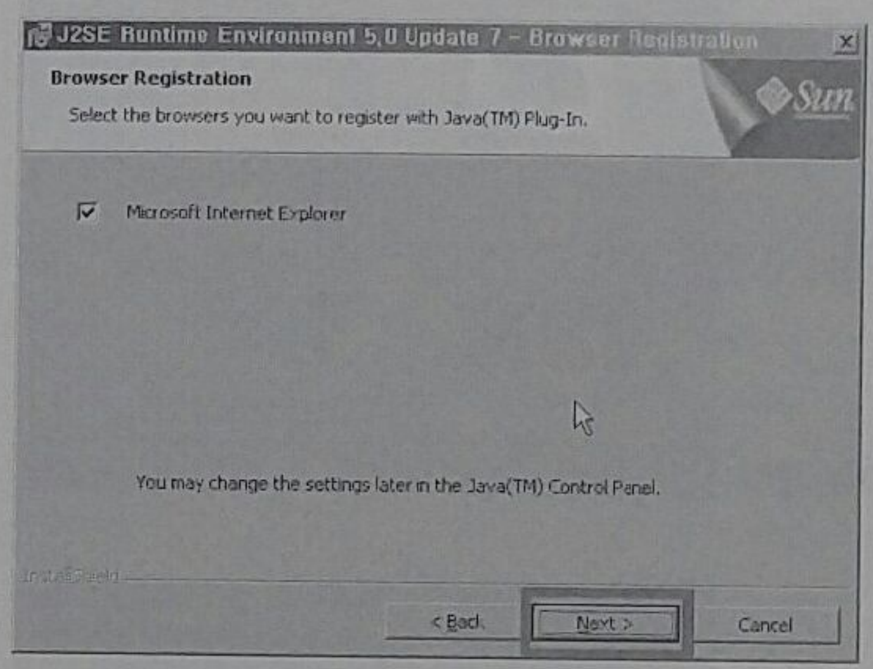
▶ 설치

⑩ 런타임 환경변수를 설치한다. next 버튼을 클릭한다.

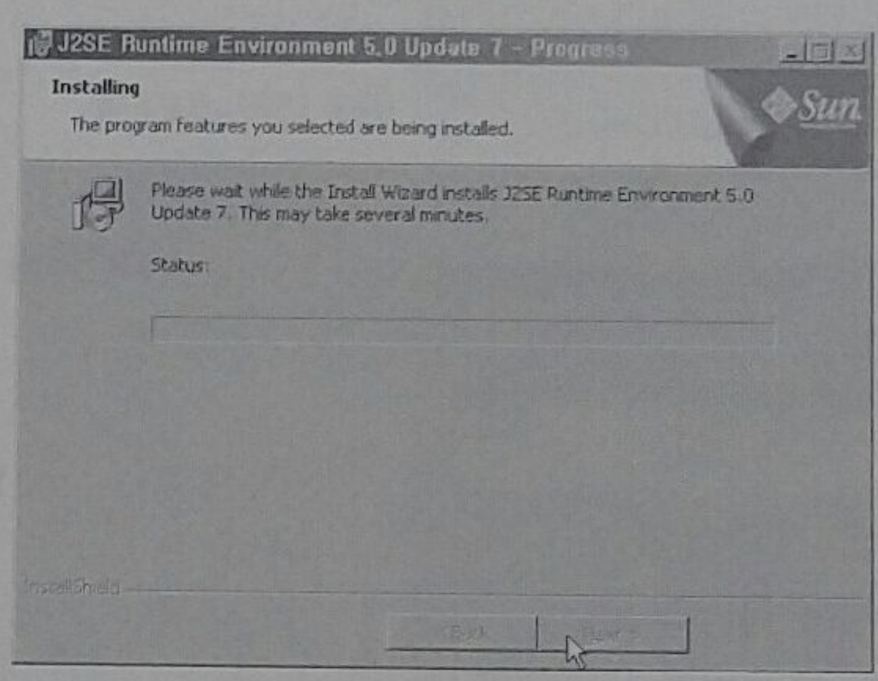


▶ 런타임 환경의 설치

⑪ 레지스트레이션 브라우저를 선택하고 next 버튼을 클릭한다.

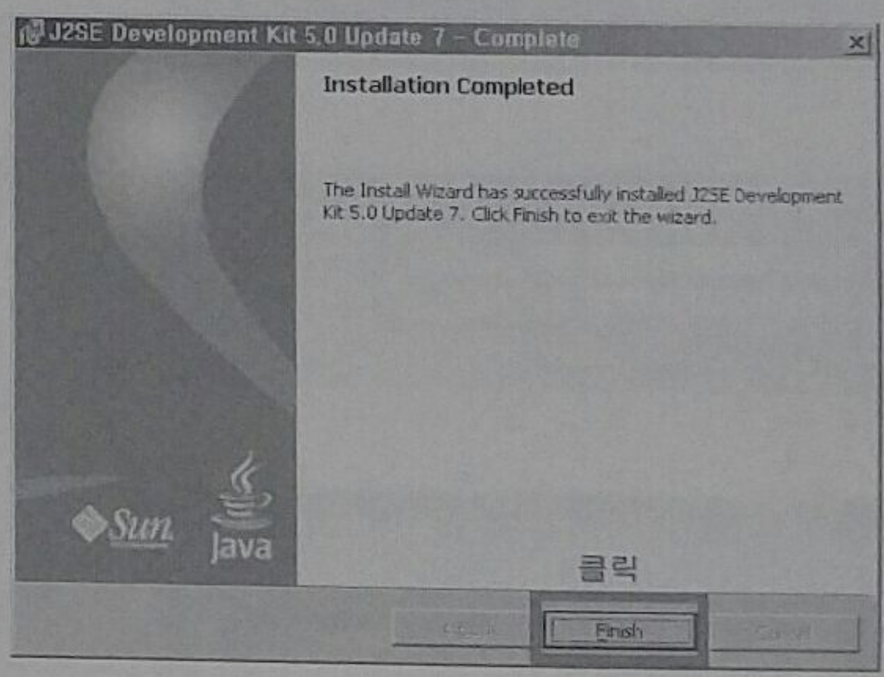


▶ 브라우저의 선택



▶ 런타임 환경의 설치

⑫ Finish 버튼을 클릭하여 설치를 완료한다.



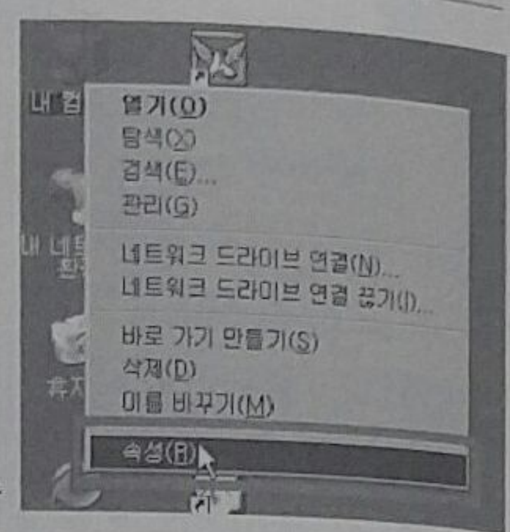
▶ 설치완료

환경설정

따라하기를 통해 jdk 설치를 하였다. 하지만 설치가 되었다고 해서 완전히 끝난 것은 아니다. jdk가 설치가 되면 설치된 jdk를 사용하는 OS에 맞게 환경변수에 등록을 해 주어야 한다.

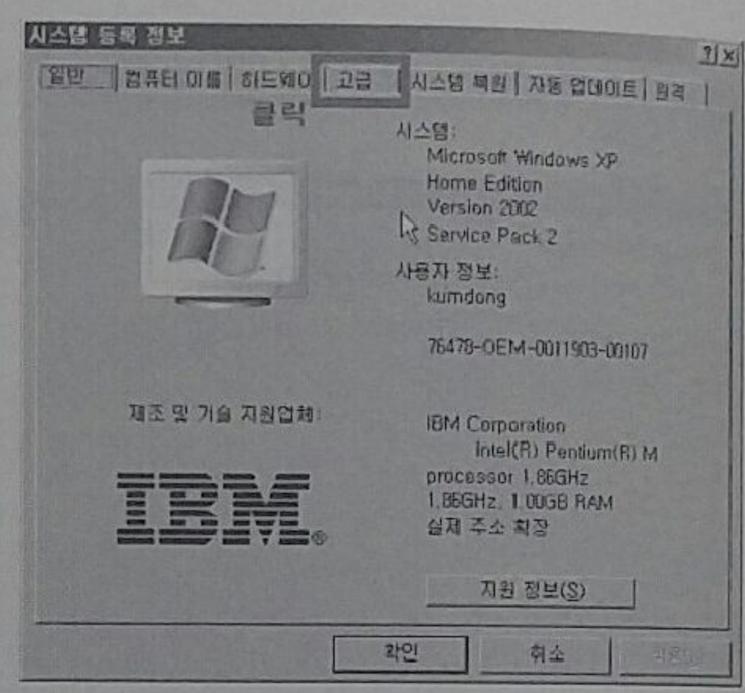
따라하기 JDK 환경설정

❶ 바탕화면의 내 컴퓨터를 선택하고 마우스의 오른쪽 버튼을 클릭하여 속성을 선택한다.



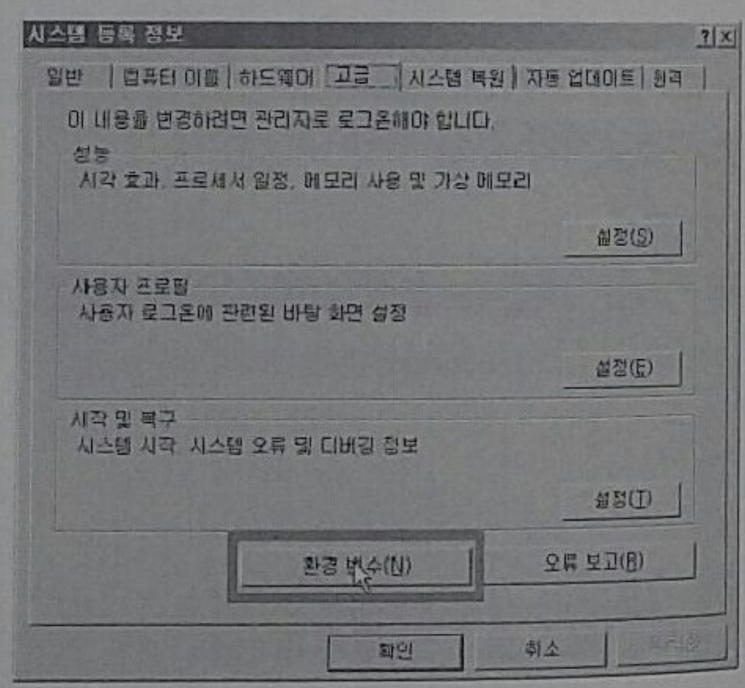
▶ 성메뉴

❷ 탭 메뉴 중에 고급을 선택한다.



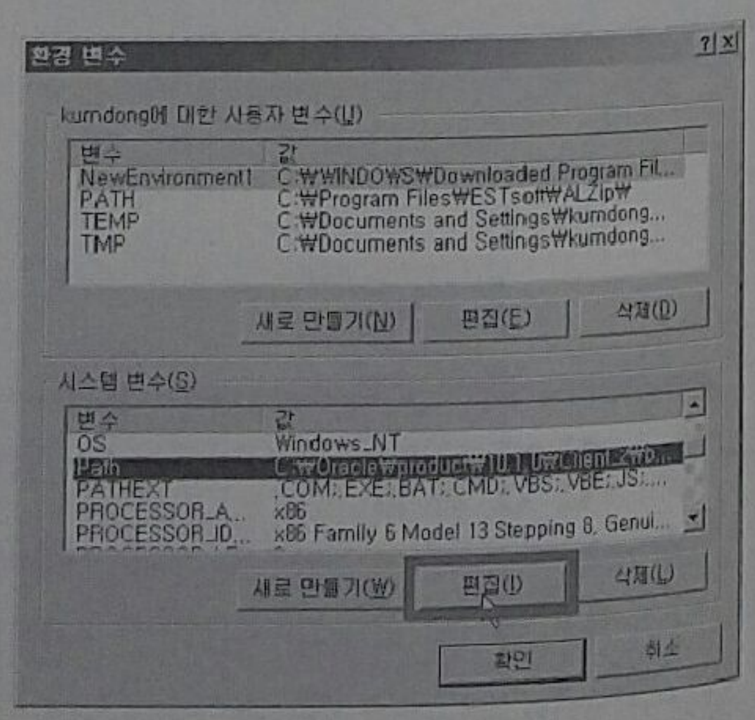
▶ 메뉴선택

❸ 환경변수 버튼을 클릭한다.



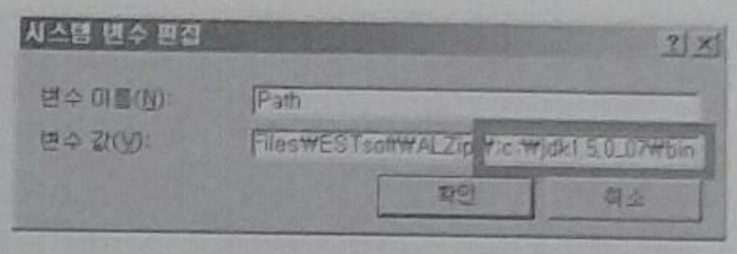
▶ 환경변수 선택

❹ Path를 선택하고 편집 버튼을 클릭한다.



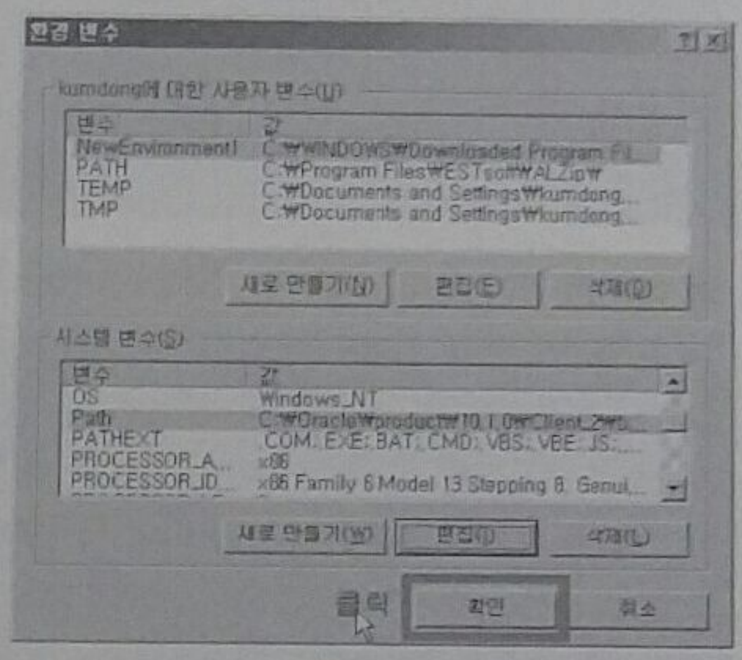
▶ Path의 편집

❺ Path의 변수 값에 설치된 jdk의 경로와 하위 폴더에 있는 bin폴더를 포함하여 입력해 준다.



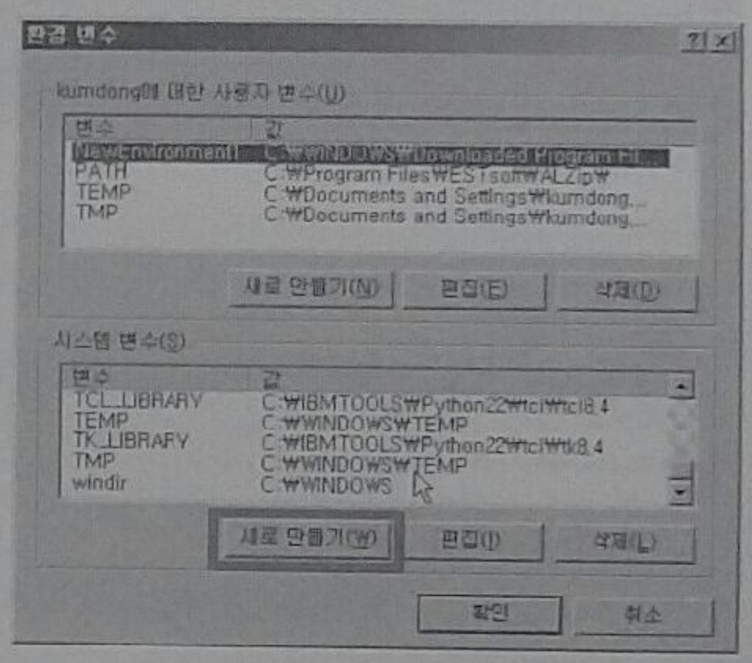
▶ 설치경로 입력

❻ ok 버튼을 누른 후 확인버튼을 클릭한다.



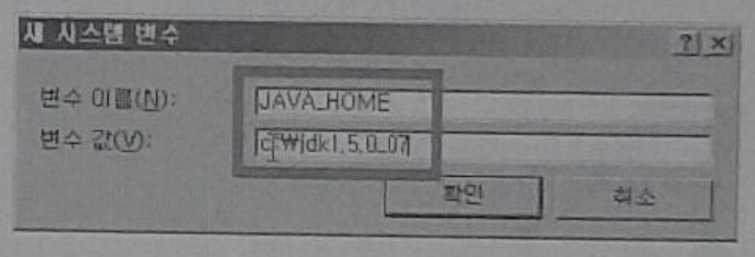
▶ Path 편집 저장

❼ JAVA_HOME을 만들기 위해 새로 만들기 버튼을 클릭한다.



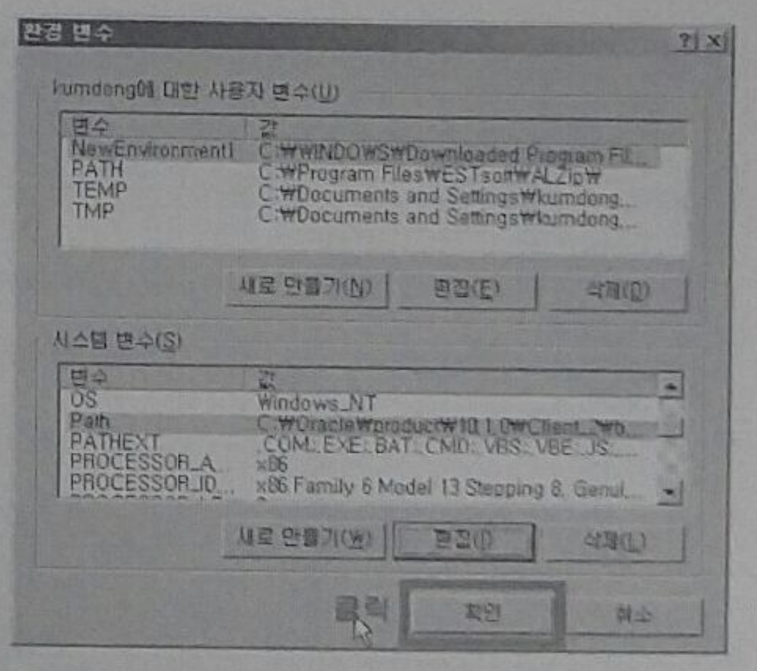
▶ JAVA_HOME 만들기

❽ 새 시스템 변수 값에 변수 이름과 값을 입력하고 확인 버튼을 클릭한다.



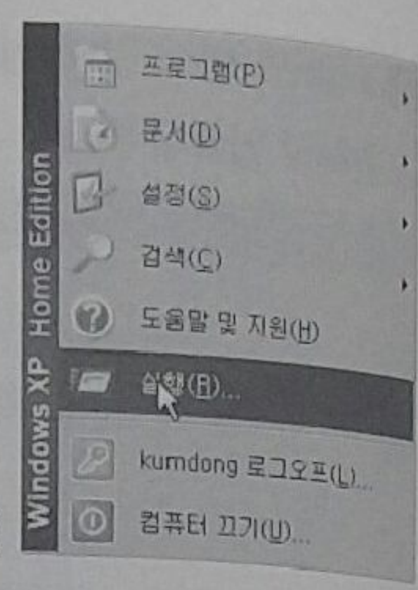
▶ 변수이름, 값 입력

❾ 확인 버튼을 클릭하여 저장한다.



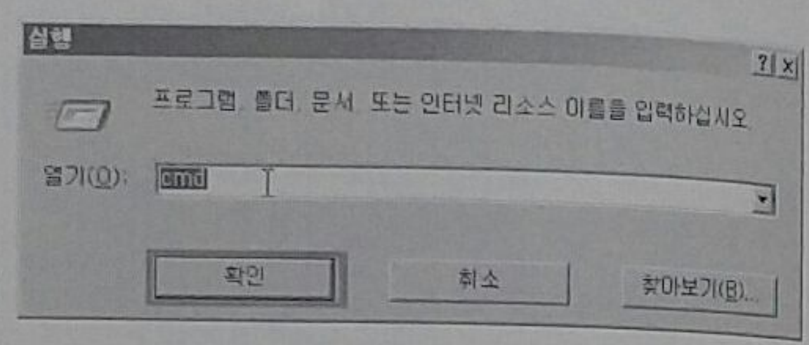
▶ 환경변수 저장

⑩ 시작의 실행 메뉴를 선택한다.



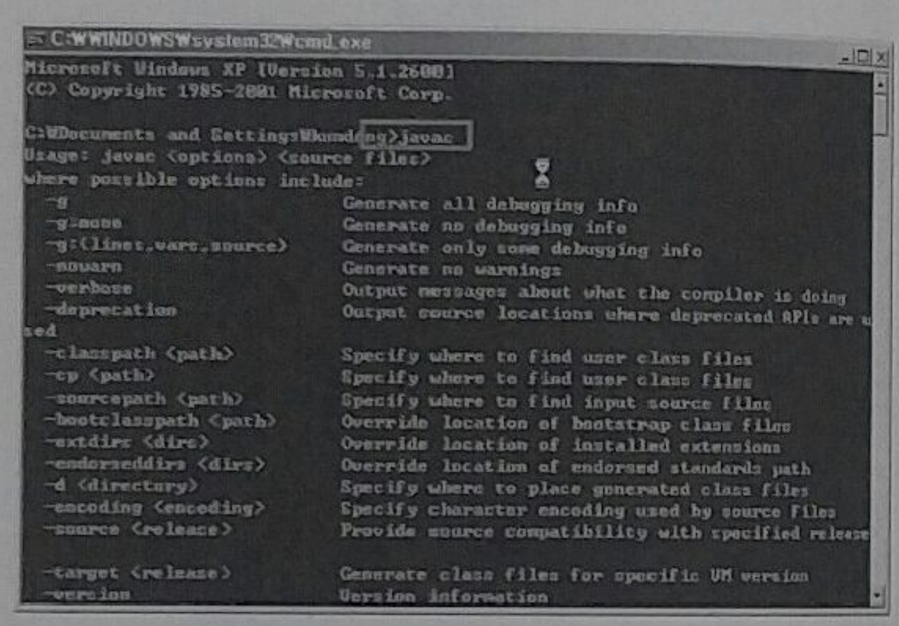
▶ 실행 선택

⑪ command 창을 띄우기 위해 cmd를 입력하고 확인 버튼을 클릭한다.



▶ command창 실행을 위한 입력

⑫ 컴파일 실행을 위한 javac을 입력하여 설정이 되었는지 확인한다.

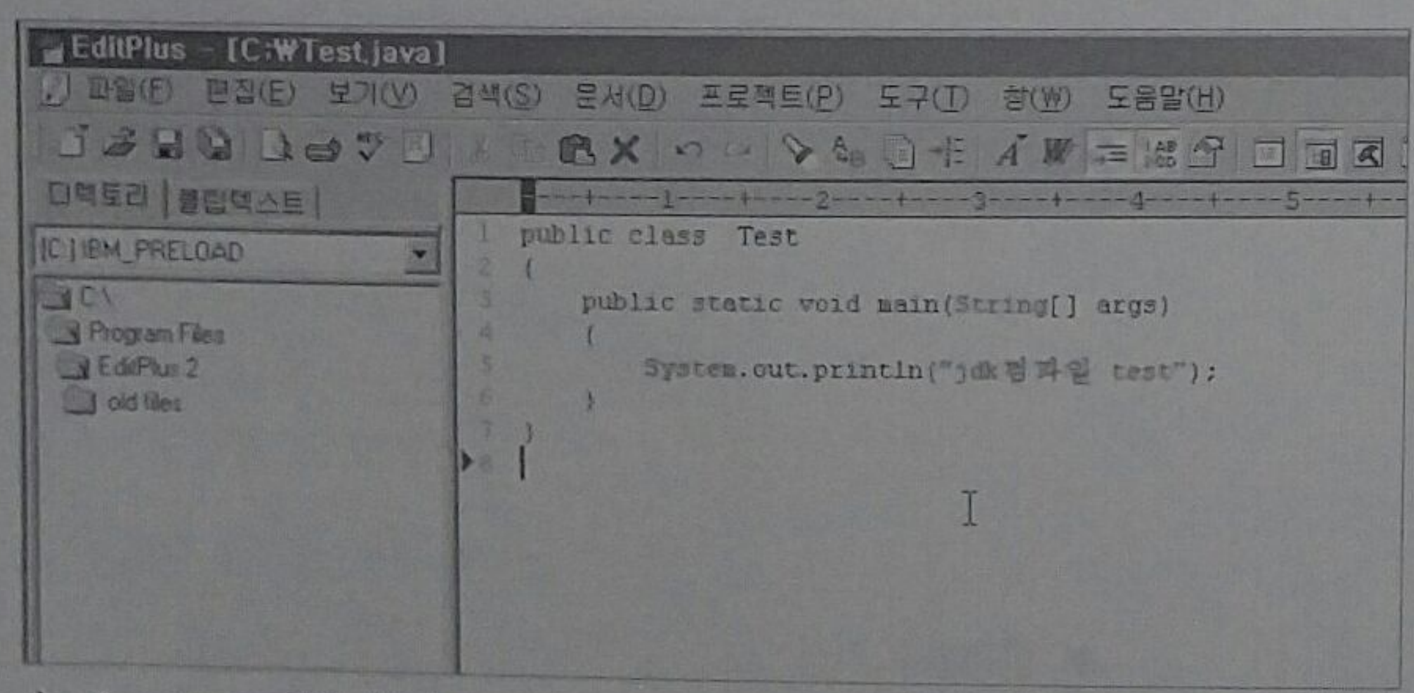


▶ 수행확인

jdk의 설치와 윈도우에서의 환경이 설정되었다. 설정이 정말로 제대로 되었는지 간단한 테스를 해 보기로 하자.

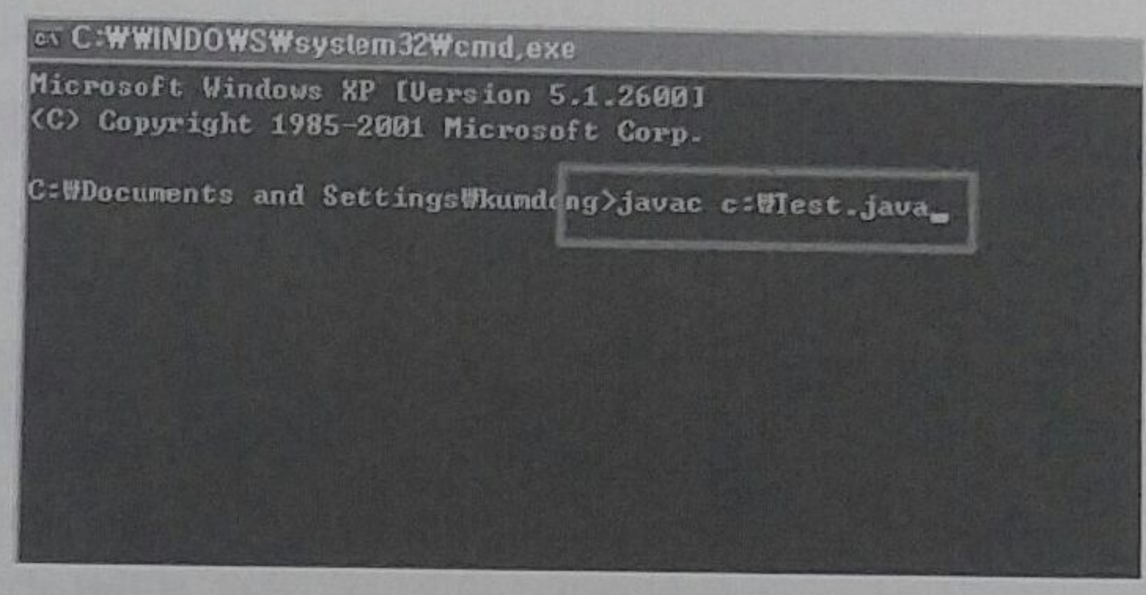
JDK 설치 테스트

❶ 에디터 툴 또는 메모장을 열어 Test.java을 작성하고 C 루트에 저장한다.



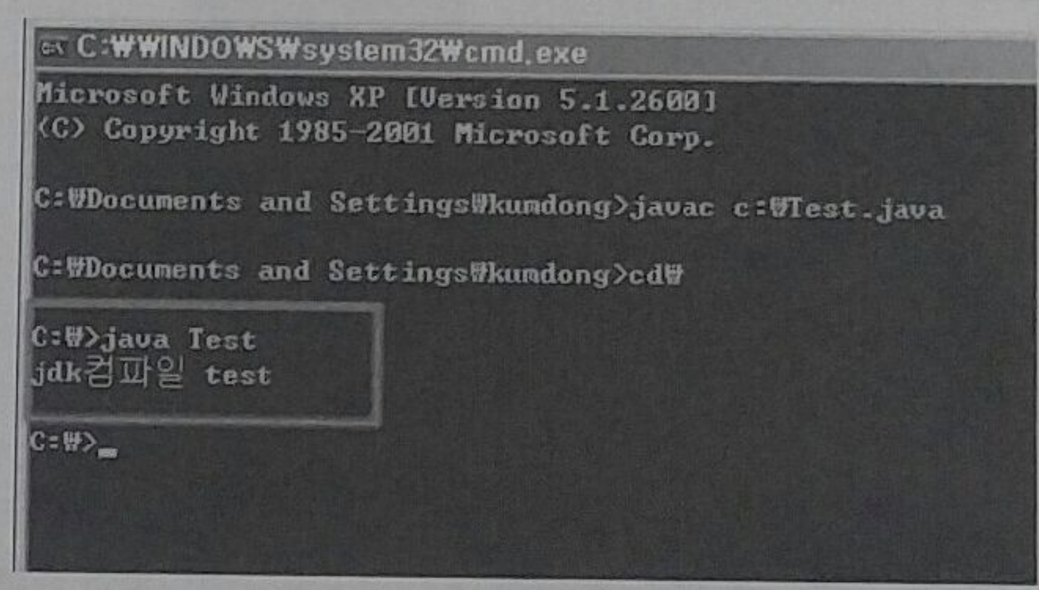
▶ Test.java 파일 작성

❷ command 창을 열고 컴파일을 한다.



▶ 컴파일

❸ 컴파일이 정상적으로 되었다면 class 파일이 있는 곳으로 이동 후 실행한다.

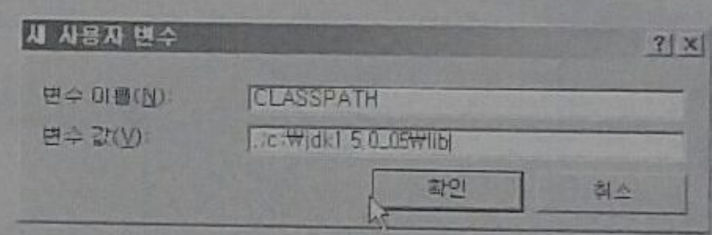


▶ Test.class의 실행

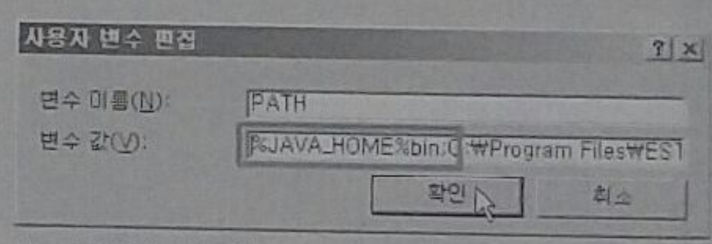
클래스 패스

jdk를 설치하고 나서 컴파일에 오류가 없음에도 불구하고 실행이 되지 않는 경우는 몇 가지가 있다. 첫 번째는 클래스 패스의 오류 부분으로 이 때 환경변수를 다시 열어 CLASSPATH 변수를 설정하고 변수 값을 주면 된다. 변수 값에 .이 들어가 있는 것을 잊지 말아야 한다.

두 번째는 기존에 사용되는 오라클 등이 있다면 Path 부분을 수정해야 한다. 위에서 JAVA_HOME이라고 선언한 부분을 환경변수 중 사용자변수 Path의 맨 앞부분에 추가한다.



▶ CLASSPATH 생성



▶ PATH수정

톰켓(Tomcat)

웹 서비스를 하기 위해서는 소프트웨어가 필요하다. 이전에는 아파치 또는 IIS가 그 역할을 해 왔지만 현재는 많은 상용 소프트웨어가 존재한다. 웹로직, 제우스 등등 고가의 웹 서버를 위한 소프트웨어는 종전에 가지는 문제점과 상호 호환성, 효율성을 개선한 아주 좋은 제품들이라고 할 수 있다.

톰켓의 경우 예전에는 독자적인 서비스를 하기 위해서는 불안정하고 효율성이 떨어진 소프트웨어였다. 개발자들을 위한 테스트용 소프트웨어라고 할까? 하지만 예전에는 그랬는지는 모르지만 현재의 시점에서 Tomcat는 아주 막강한 파워를 자랑하고 있다. 중소기업 또는 일반 쇼핑몰 등등 독자적인 웹 서

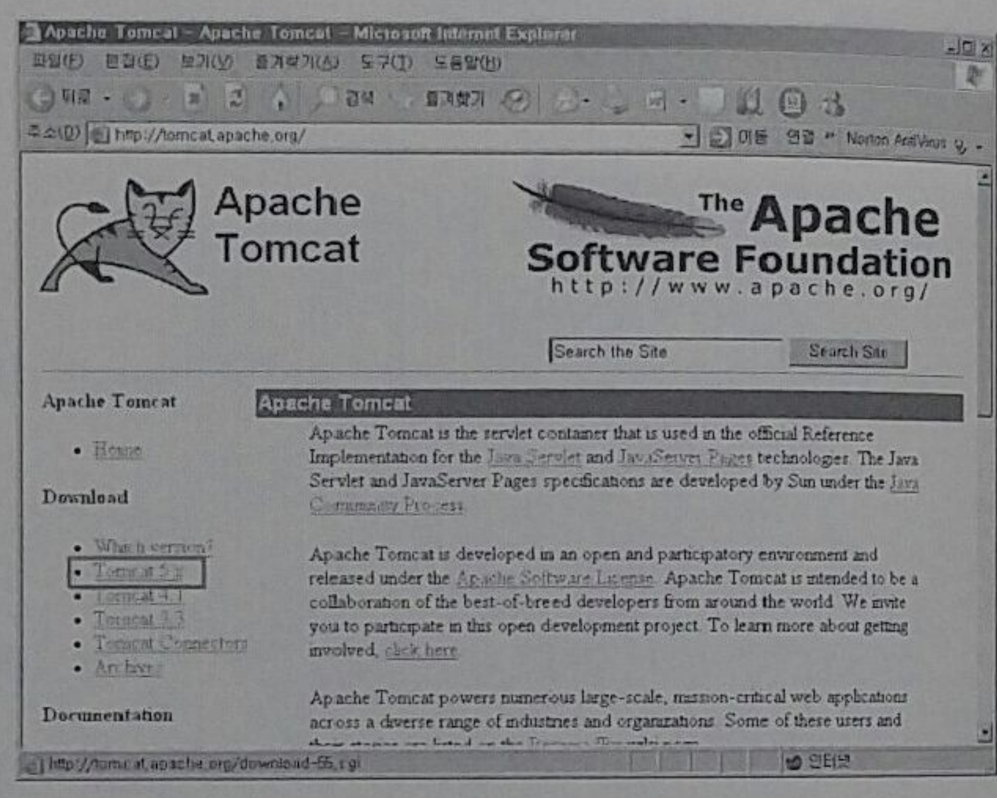
비스를 위해 사용해도 무리가 없을 정도로 많은 발전이 이루어졌다. 실제로 필자가 개발한 프로그램도 톰켓을 기반으로 하여 개발하였고 사용자가 상당했지만 문제가 없었다. 앞으로도 톰켓은 더 많은 발전과 많은 개발자로부터 사랑을 받을 것이다.

톰켓 설치하기

도서출판 대림 홈페이지 자료실에서 다운로드한 apache-tomcat-5.5.17.exe를 실행 또는 tomcat.apache.org를 방문하여 다운로드 하여 설치하여도 무방하다.

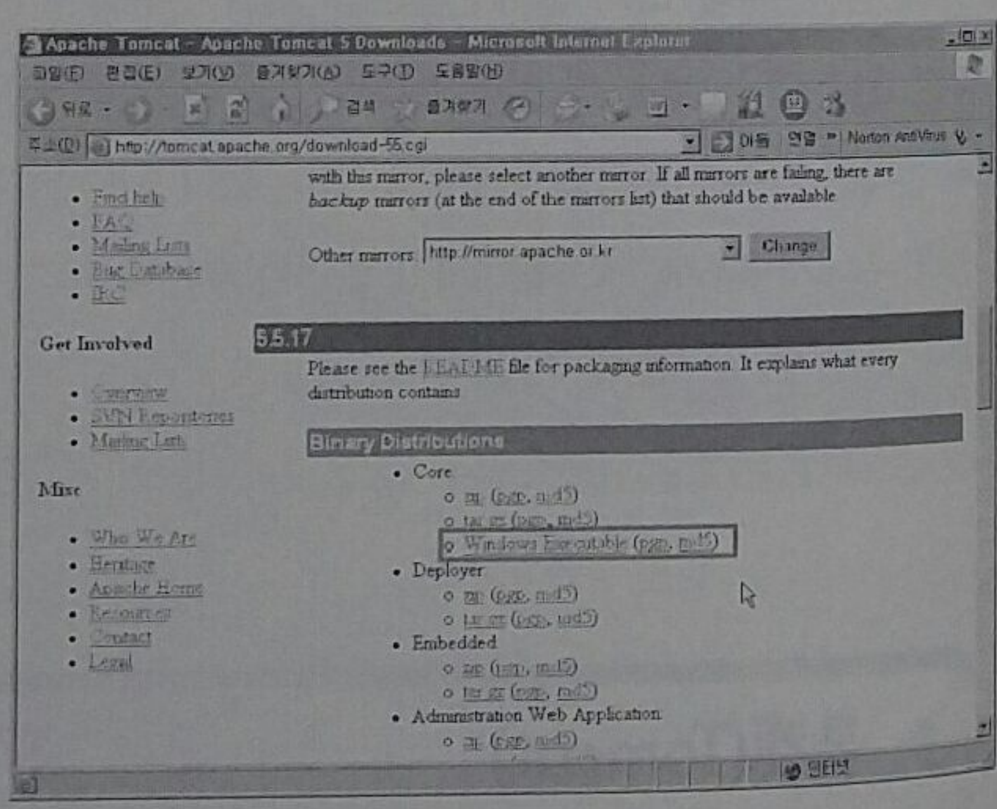
따라하기 톰캣 설치

❶ tomcat.apache.org에 접속하여 Download 메뉴에서 Tomcat5.X를 선택한다.



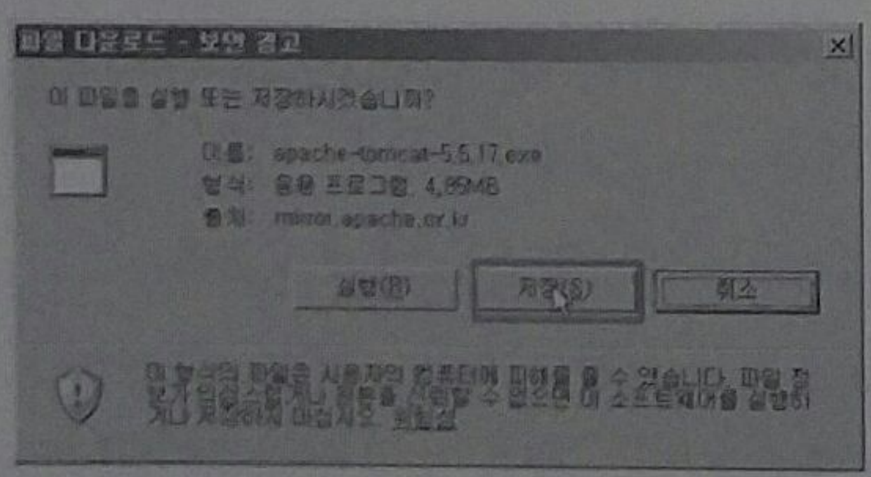
▶ tomcat.apache.org 접속

❷ 사용하고자 하는 OS에 맞게 다운로드하면 된다. 윈도우용을 선택한다.

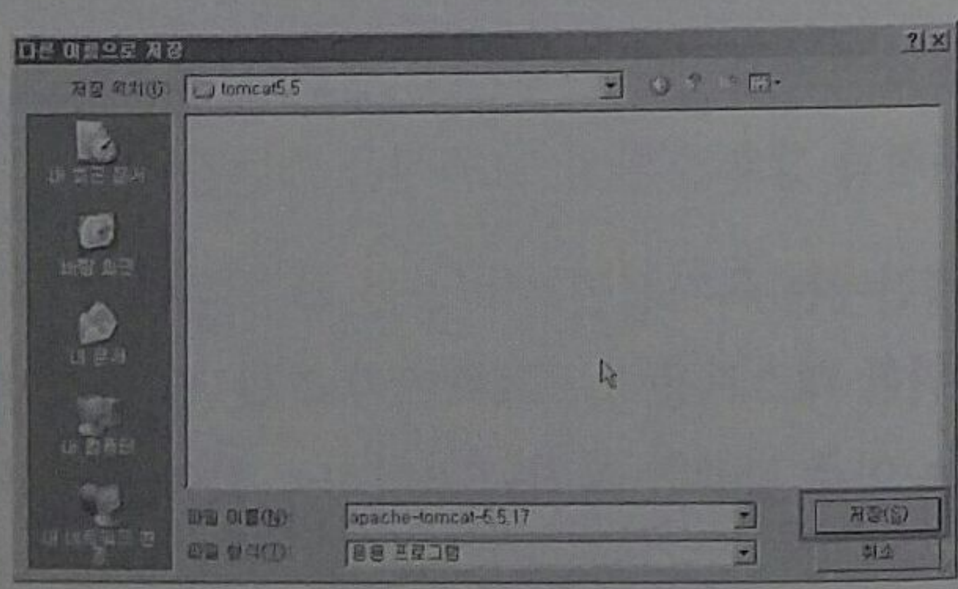


▶ 윈도우용 Tomcat

❸ 다운받을 파일을 사용자가 지정한 폴더에 저장한다.

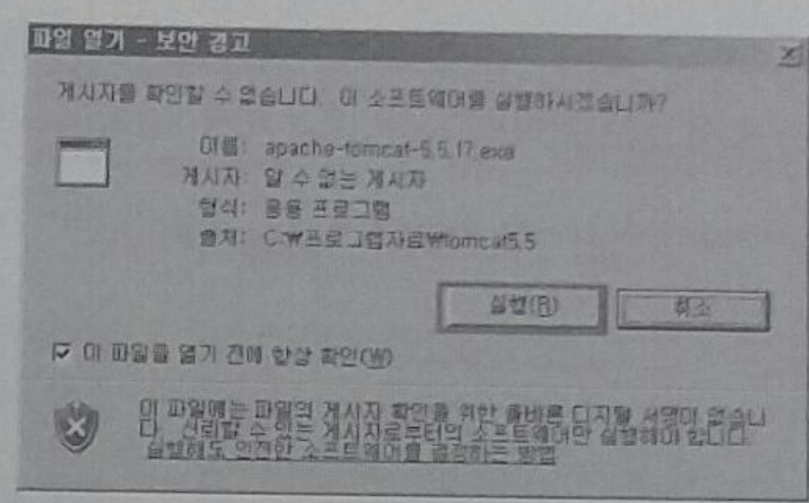


▶ 파일다운



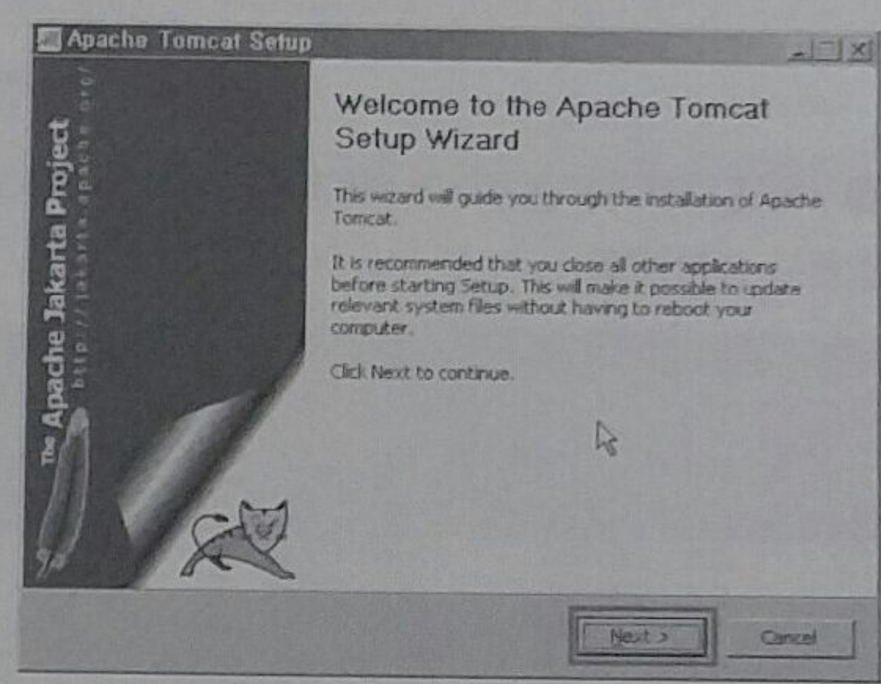
▶ 파일의 저장

❹ 다운이 되었으면 더블클릭하여 실행한다.



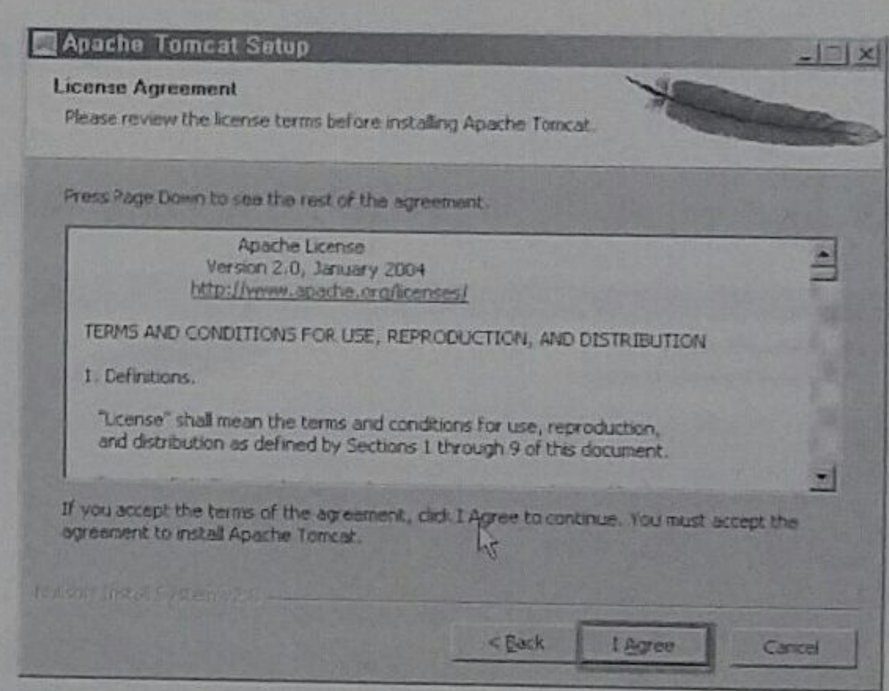
▶ 실행

❺ Next 버튼을 클릭한다.



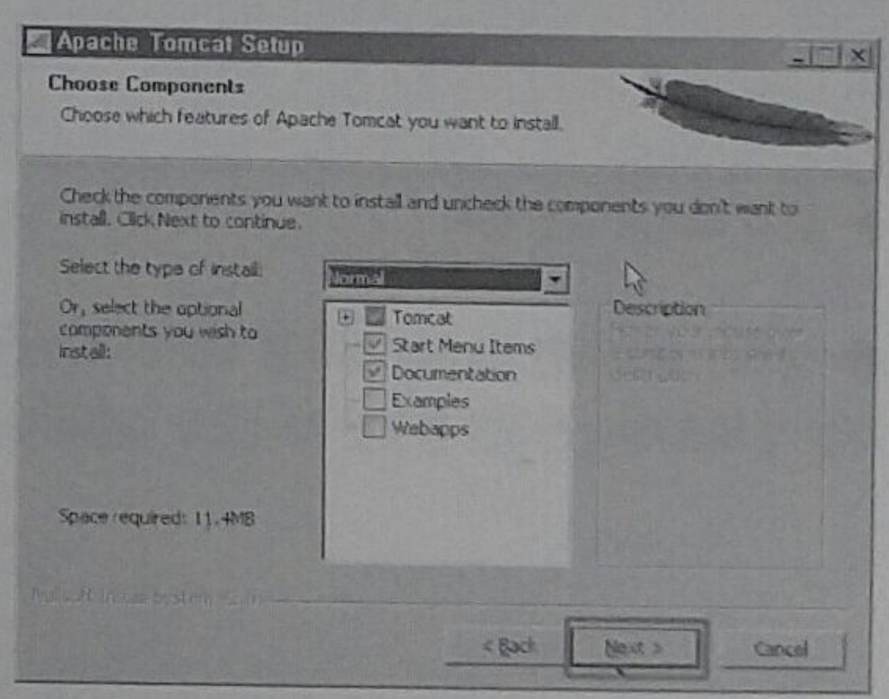
▶ Tomcat 마법사 실행

❻ 'I Agree' 버튼을 클릭하여 라이센스에 동의한다.



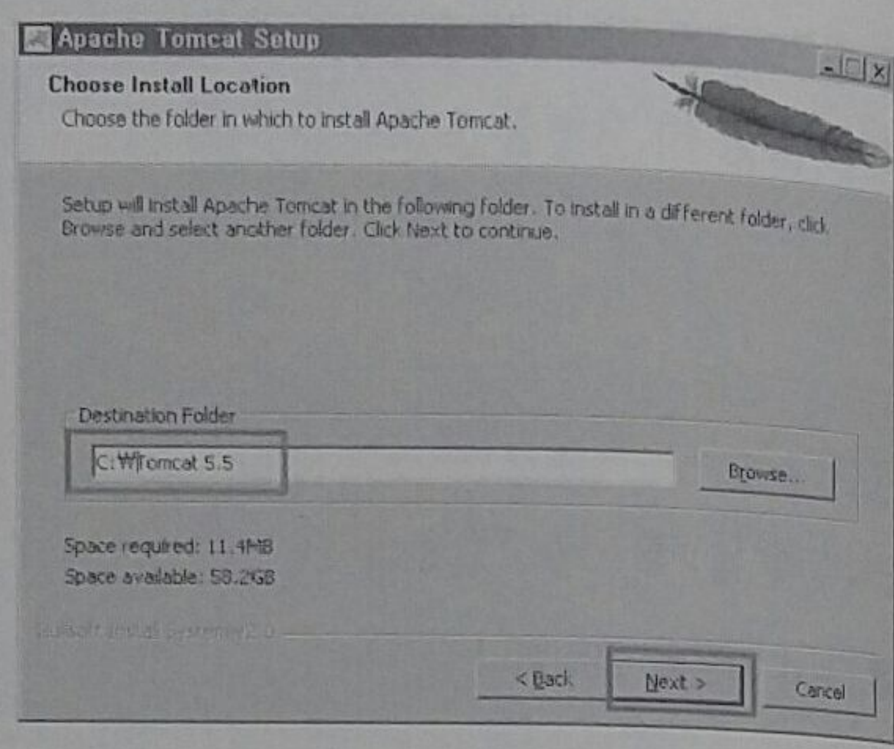
▶ 라이센스에 동의

❼ Tomcat의 사용자 설치 설정으로 Next 버튼을 클릭한다.



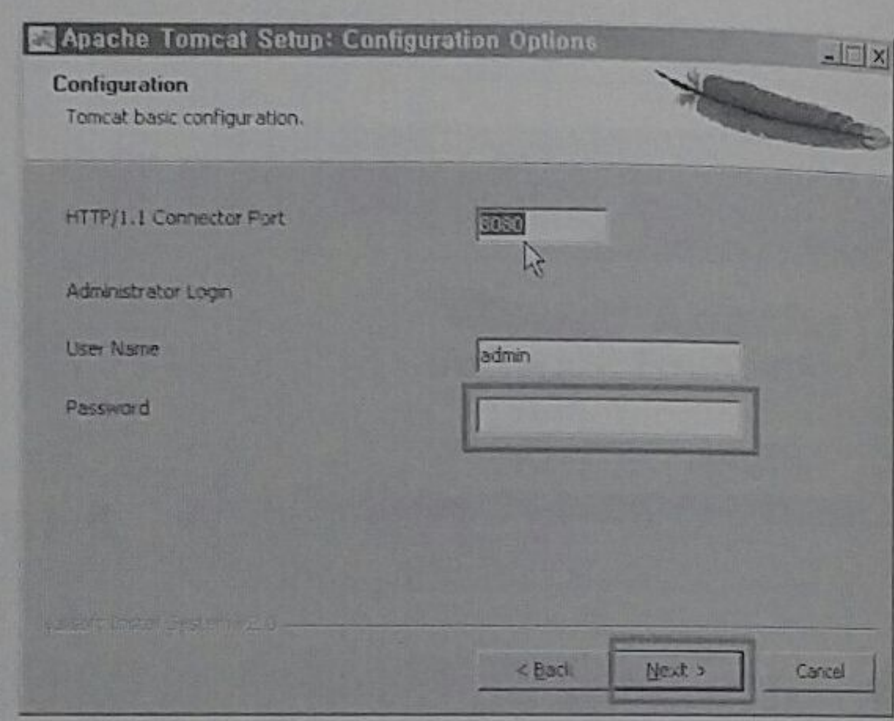
▶ 설치설정

❽ 설치할 폴더 찾기를 통해 지정하던지 아니면 바로 입력한 후 Next 버튼을 클릭한다.



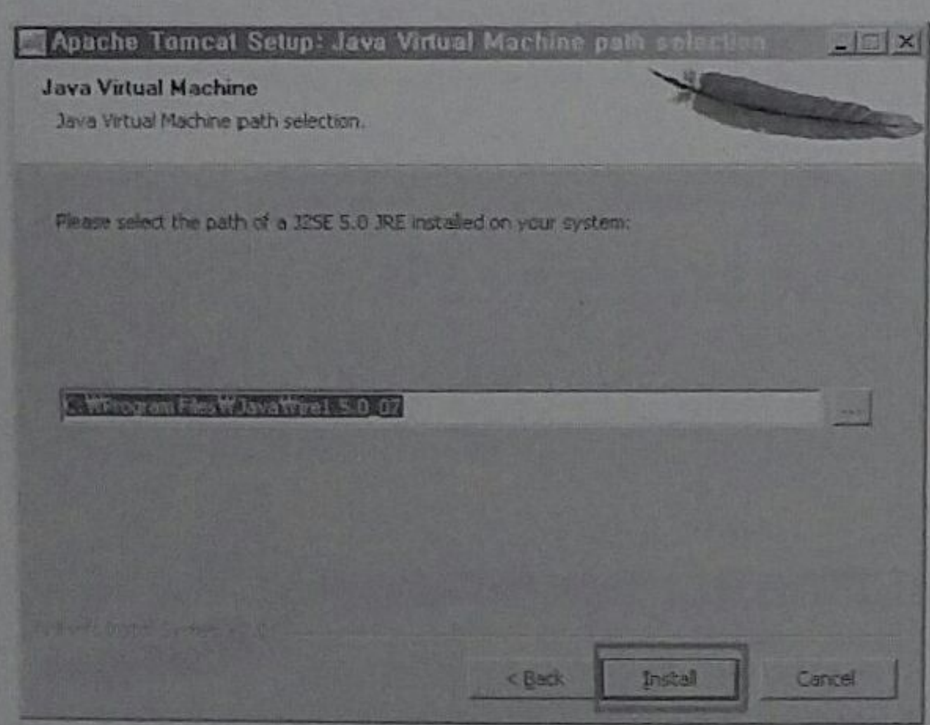
▶ 설치폴더 지정

❾ 사용할 포트와 관리자의 아이디 암호를 입력한다.[기본적으로 암호만 설정한다]

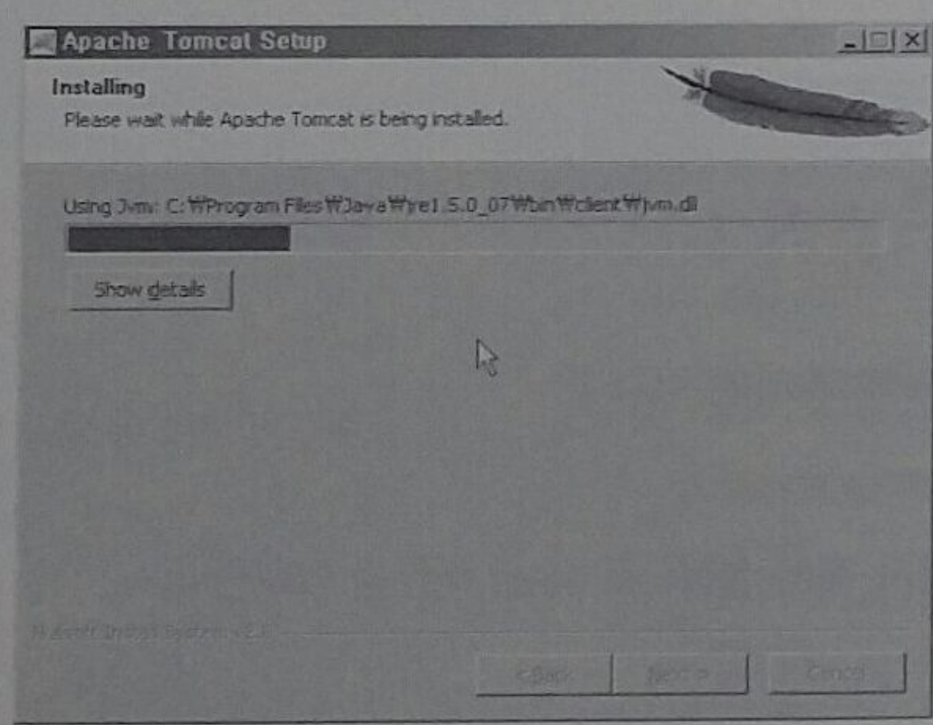


▶ 포트 및 관리자 입력

❿ Java Virtual 머신의 설치하기 위해 install 버튼을 클릭한다.

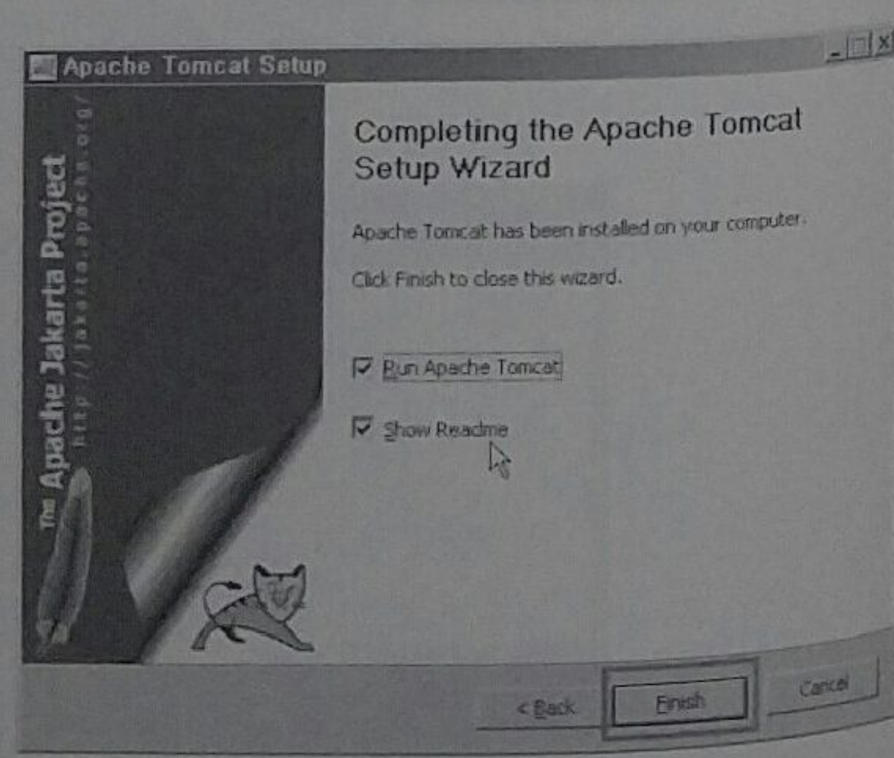


▶ Java Virtual 머신 설치



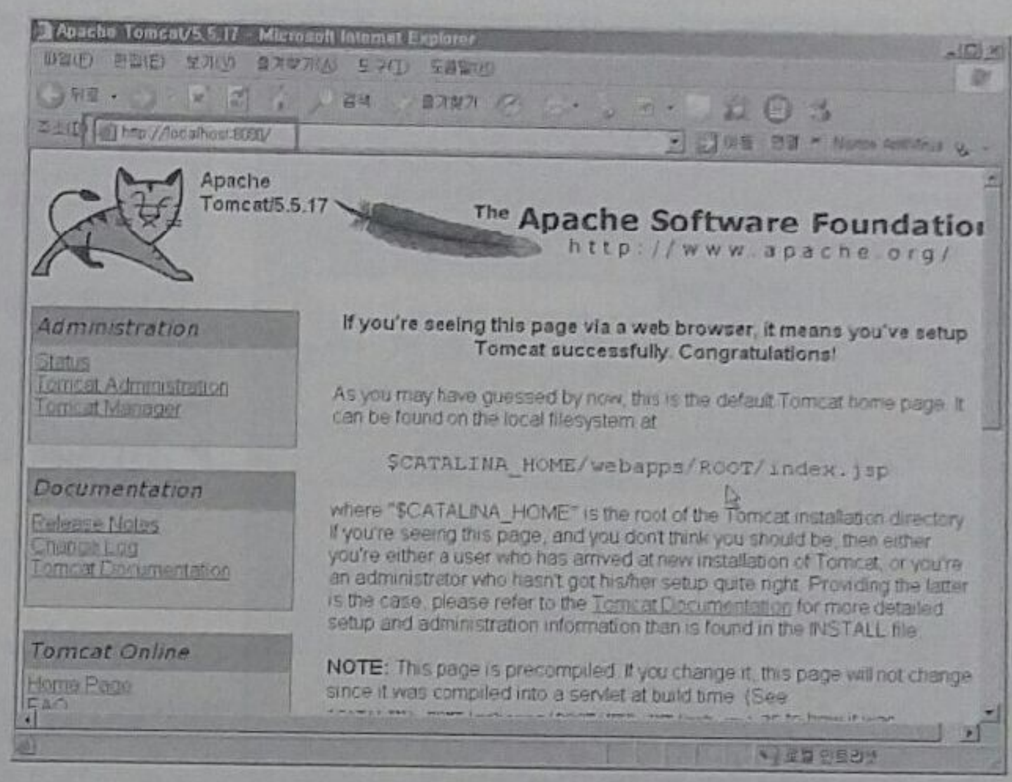
▶ 설치중

⓫ 설치가 완료되면 Finish 버튼을 클릭한다. 만약 체크박스의 부분이 체크가 되었다면 톰켓은 자동으로 실행된다.



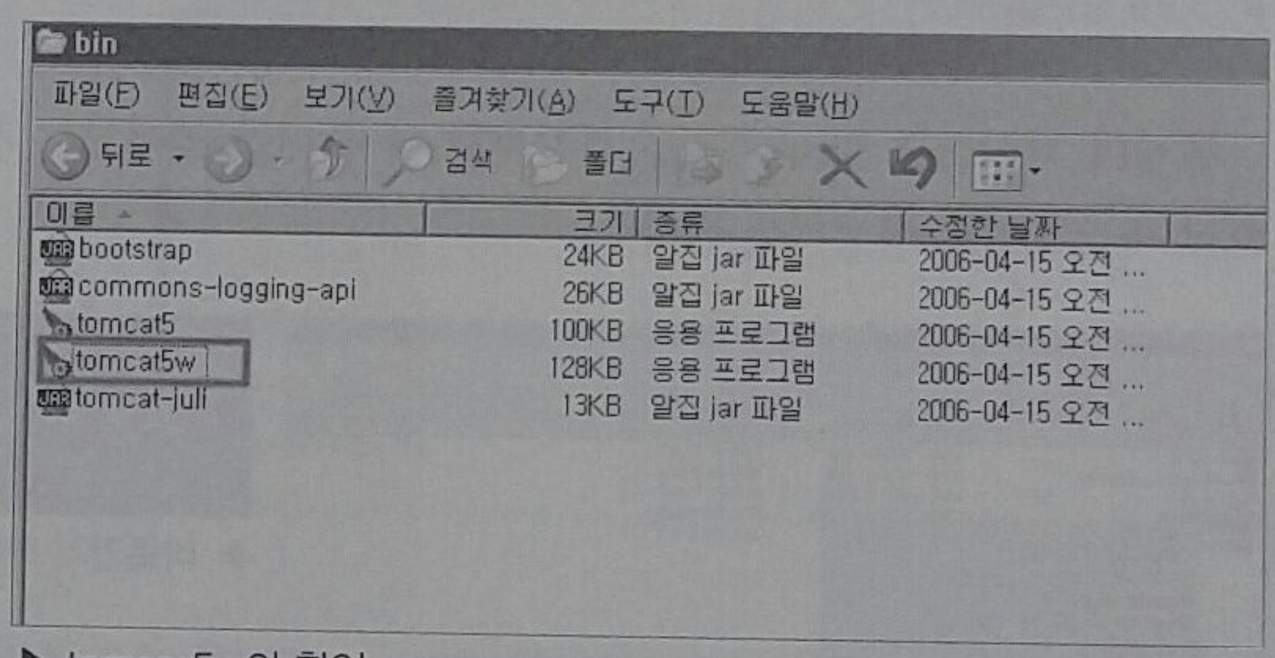
▶ 설치완료

⓬ 설치 후 자동실행으로 체크가 되어 있음으로 브라우저를 열고 톰켓이 구동되는지를 확인한다.



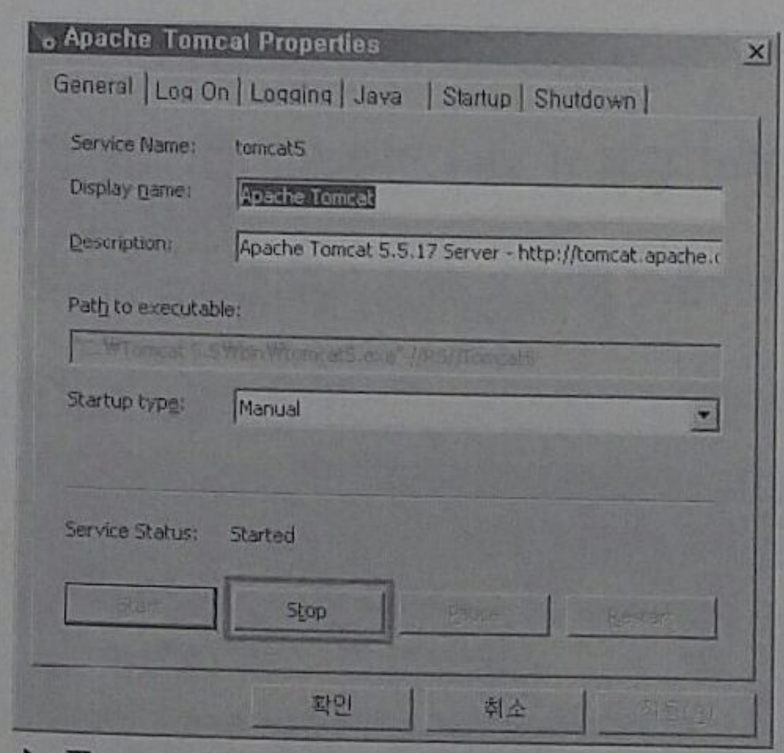
▶ Tomcat의 구동 확인

설치가 되어 브라우저에서 고양이의 그림이 나왔다면 제대로 설치가 되었고 구동이 된 것이다. 만약 실행중인 톰켓을 중지하고자 한다면 설치된 톰켓의 폴더의 하위 폴더인 bin폴더에 tomcat5w 파일을 먼저 확인한다.

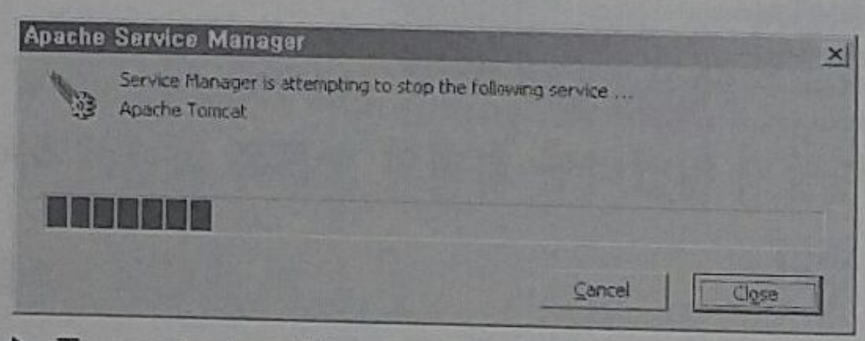


▶ tomcar5w의 확인

더블클릭하면 Tomcat 속성 창이 출력된다. 현재 서비스중이라면 Stop 버튼을 클릭하여 톰캣을 중지시킨다.

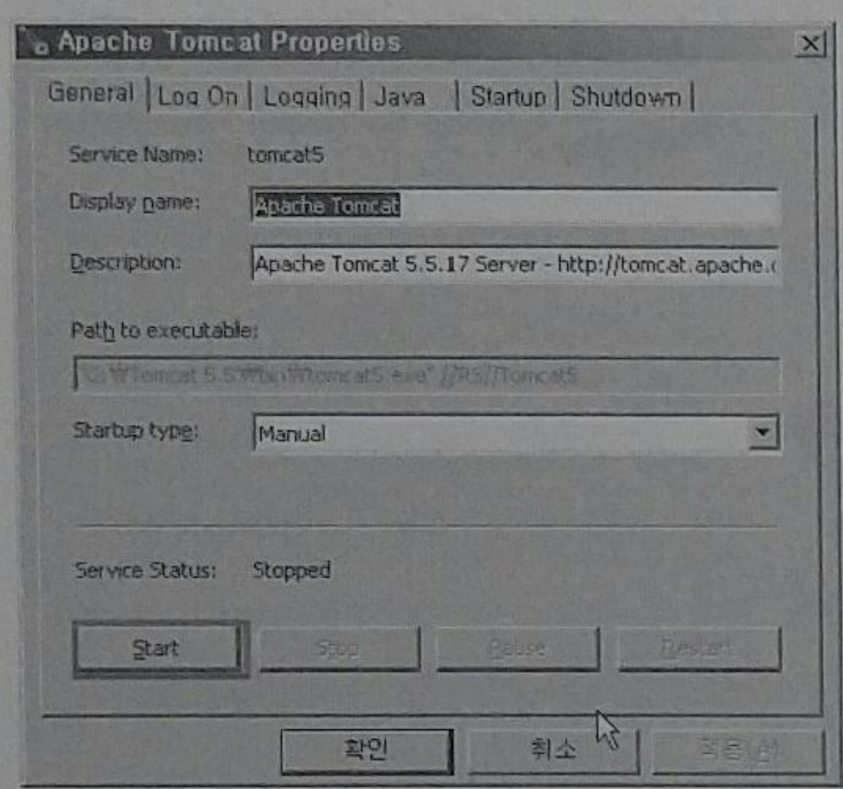


▶ Tomcat stop

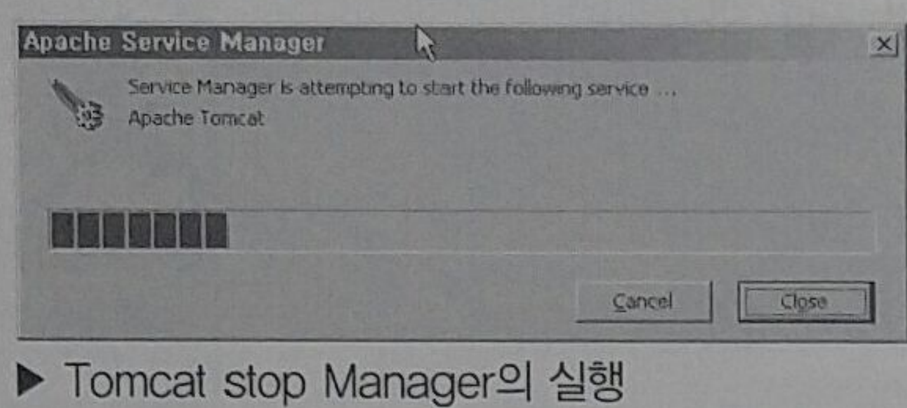


▶ Tomcat stop Manager의 실행

다시 톰켓를 실행하고자 한다면 Start 버튼을 클릭하면 된다.

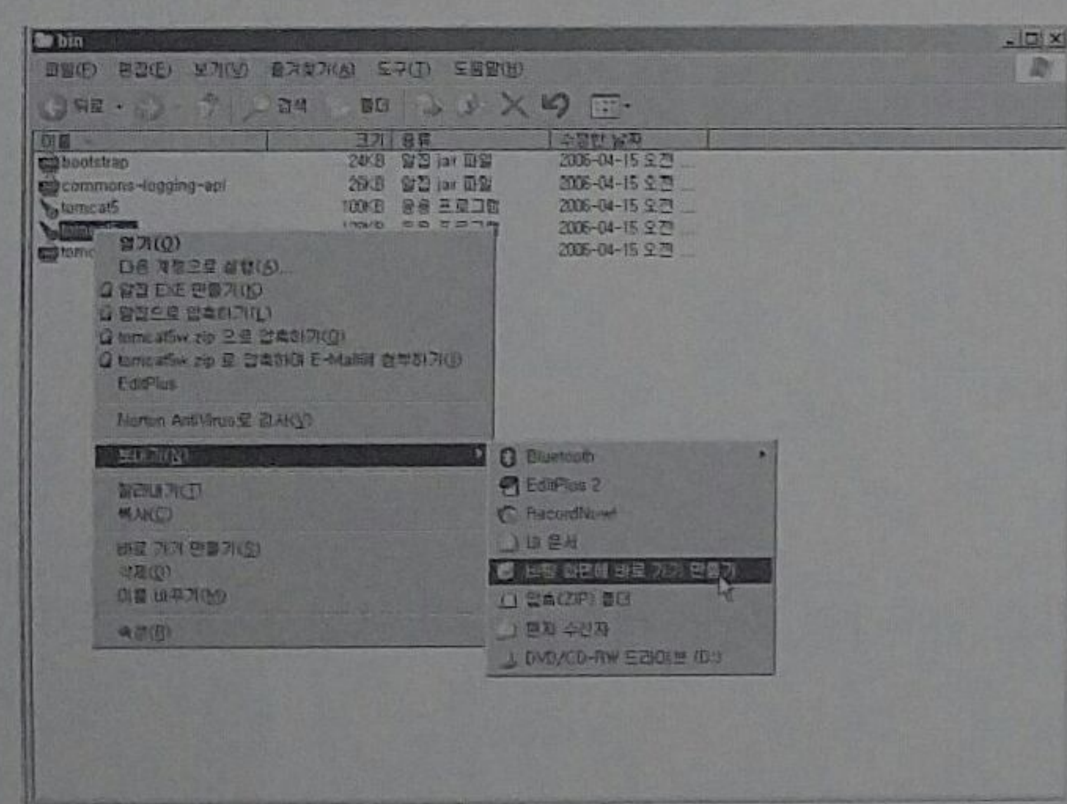


▶ Tomcat Start

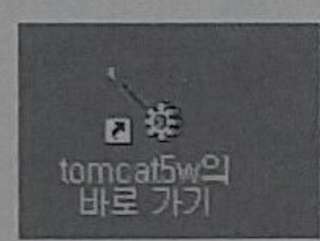


▶ Tomcat stop Manager의 실행

톰켓의 서비스를 Start/Stop 시킬 때마다 폴더를 열어보는 것이 번거롭다면 tomcar5w를 바탕화면으로 바로가기로 생성하면 된다.



▶ tomcar5w 바로가기 만들기

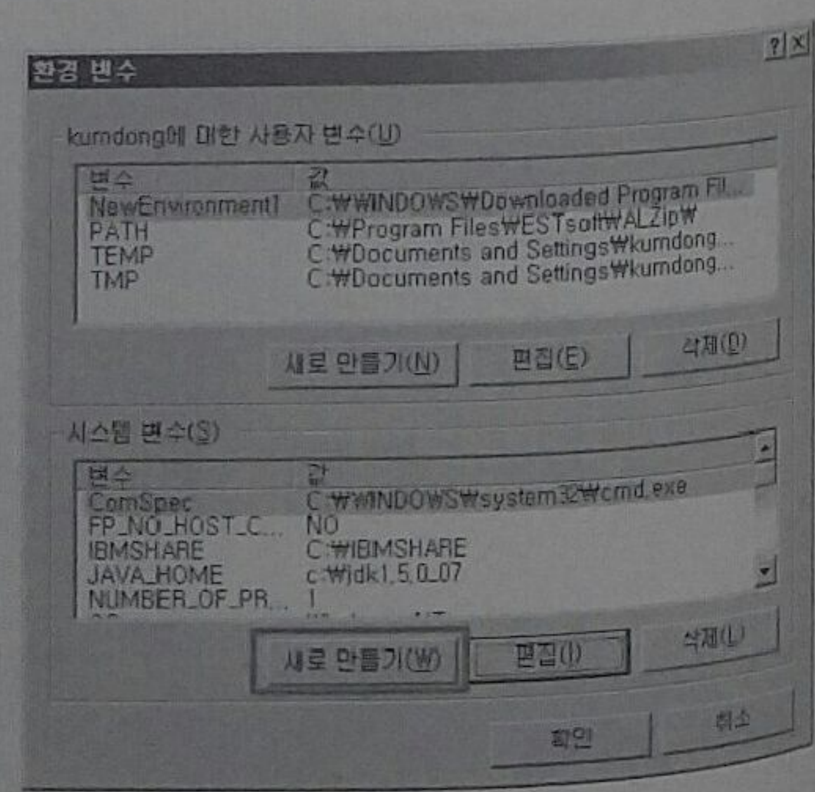


▶ 바로가기 아이콘 생성

만약 바로가기를 만들지 않았다면 화면의 오른쪽 하단에 아이콘이 등록이 되어 있음으로 그것을 사용해도 된다.

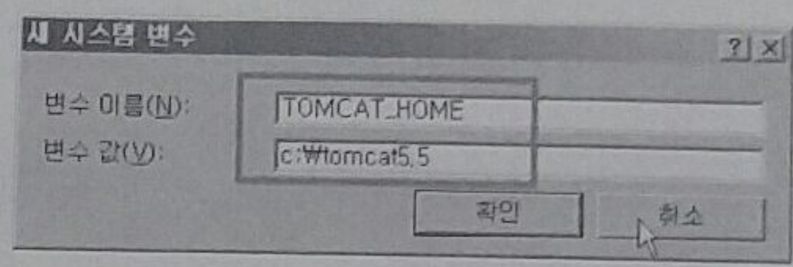
▶ tomcar5w의 생성된 하단 아이콘

톰켓도 JDK와 마찬가지로 TOMCAT_HOME을 만들어 준다. 환경변수 입력창을 통해 새로 만들기 버튼을 클릭한다.



▶ 환경변수 창

변수명과 변수 값을 주고 확인 버튼을 클릭하면 된다.

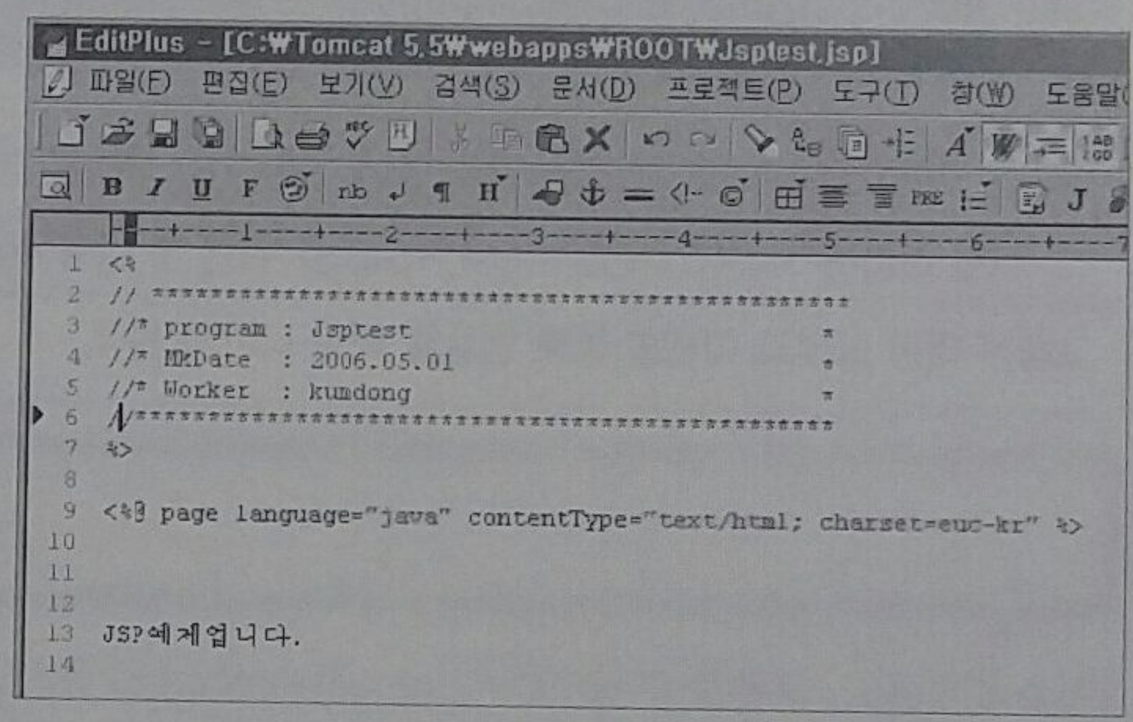


▶ TONCAT_HOME 지정

이로서 JDK와 TOMCAT의 설치가 되었다. 설치된 JDK와 Tomcat이 제대로 구동이 되고 브라우저를 통해 구현이 되는지 따라하기를 통해서 살펴보기로 하자.

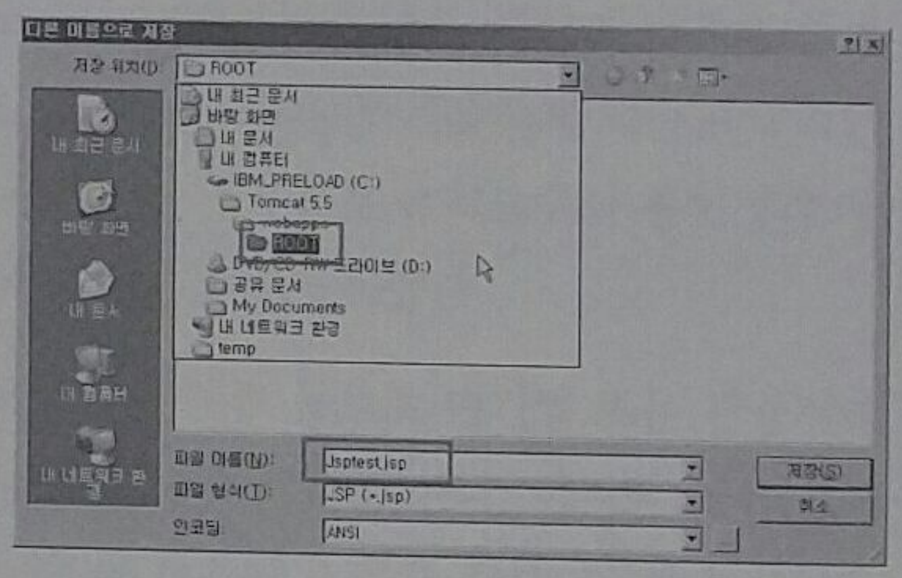
따라하기 JDK와 TOMCAT 연동 확인

❶ 에디터를 통해 간단한 JSP를 작성한다.



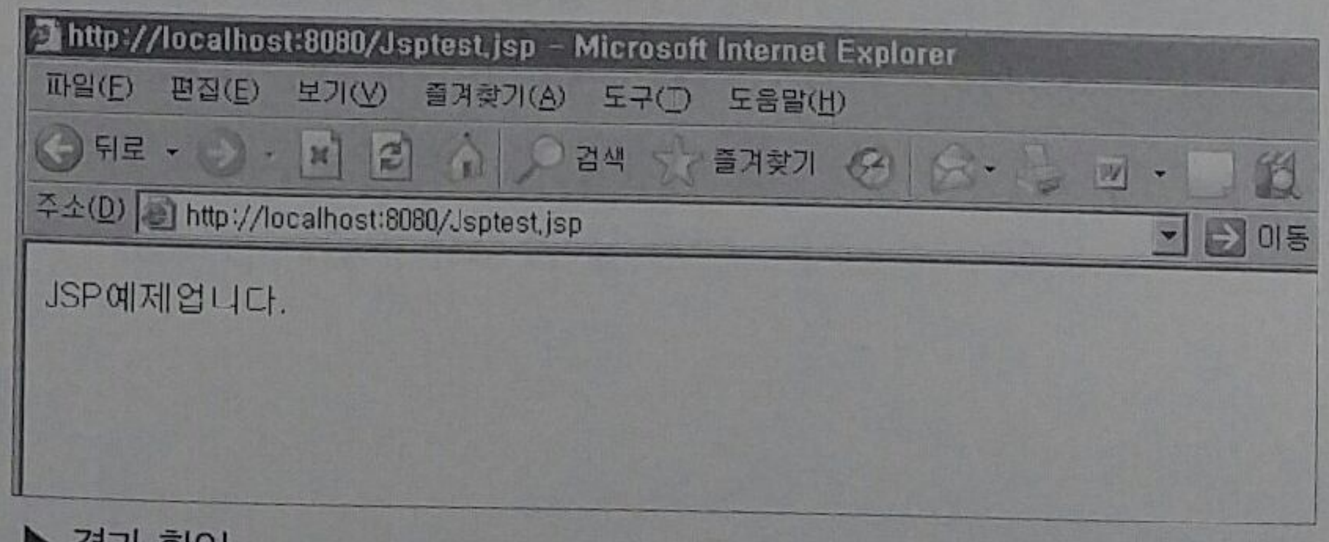
▶ Jsptest.jsp

❷ 작성된 파일을 저장한다. 저장 시에는 기본적으로 톰켓의 Root 폴더에 저장을 한다.



▶ Jsptest.jsp의 저장

❸ 브라우저를 열고 Jsptest.jsp를 호출하면 출력이 보인다.



▶ 결과 확인

다음 부분은 기억해 두기로 하자. JSP 프로그램을 작성할 시에는 이 부분이 꼭 들어간다.

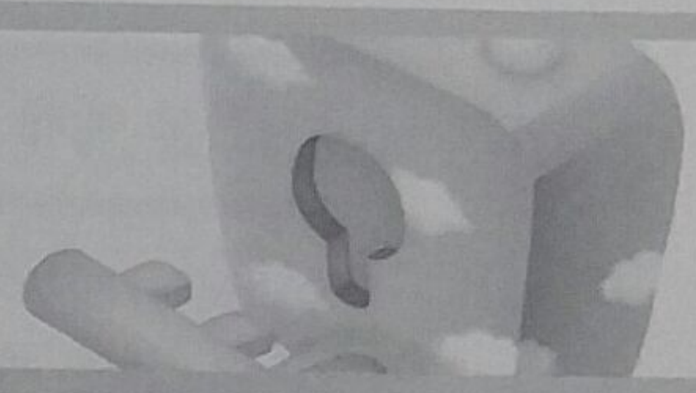
```
<%@ page language="java" contentType="text/html; charset=euc-kr" %>
```

이것의 의미는 현재 페이지의 언어는 java이고 보여주는 타입은 텍스트와 HTML 형식으로 보여주며 출력되는 문자는 한글로 표현된다는 것이다. 만약 이 부분이 없거나 한다면 굳이 에러를 발생시키지는 않지만 브라우저에 표현되는 부분이 많이 달라질 것이다.

맺음말

우리는 이번 장에서 웹 프로그래밍을 하기 위한 가장 기초적인 JDK의 설치와 Tomcat의 설치에 대해서 살펴보았다. 이제 독자는 TMJ(Tomcat+Mysql+Jsp)를 위한 첫 발걸음을 시작한 것이다. 다음 장에서는 본격적인 JSP 프로그램에 대한 공부와 예제를 통해 팁들을 배워 나가게 된다.

08 TMJ의 시작



안녕하세요

이번 장에서는 TMJ(Tomcat+Mysql+Jsp)의 기초적인 부분부터 시작한다. java가 가지는 무수히 많은 메소드 등에 대해서는 이 책에서는 열거하지 않는다. 철저하게 기본을 숙지해 가면서 하나 둘씩 팁을 익혀 나가는 방향으로 공부할 것이다. TMJ를 숙지하고 나면 TMJ의 프로그램에 대한 자신감을 가지게 될 것이다.

TMJ를 사용하자

많은 웹에이전시 또는 기타 중소 IT 쪽에서 AMP(Apache+Mysql+PHP)를 사용한다. AMP를 많이 사용했던 이유 중에 가장 중요한 것은 비용이다. 작은 쇼핑몰을 인터넷으로 운영하고자 한다면 장비 + S/W가격 + 개발인건비 + 유지보수 등등 많은 비용이 들어간다. 따라서 사업주는 사업의 비용을 줄여 마케팅 비용이나 기타 다른 부분에 투자하고 싶어 할 것이다. 그러기에 AMP를 사용하게 되면 S/W적인 비용은 추가적으로 발생하지 않는다. 게다가 PHP가 가지는 개발의 편리성과 유지보수의 용이성과 응답속도가 빠른 장점으로 많은 중소업체들이 선호하게 되어 빠른 발전을 이루어졌다.

이에 반해 Java 계열의 서블릿이나 JSP 등은 개발하기 어렵다는 이유 등으로 대기업 또는 관공서에서만 특정하게 사용되고 개발되어졌기 때문에 대중화가 힘들지 않았나 생각을 한다. 하지만 이제는 다르다. 예전에는 그러했는지 모르지만 많은 개발자가 자신의 Tip를 오픈하고 상호 커뮤니티를 형성함으로서 비약적인 발전을 가져왔다.

TMJ를 사용함으로서 S/W적인 비용이 들어가지 않으며 java가 가지는 객체화 부분이 결합하여 더 안정적이고 막강한 기반이 마련되었다는 것이다. 하지만 아쉬운 것은 현존하는 중소업체들은 기존에 개발되어졌고 운영되어진 또는 새로이 개발하고자 모든 부분을 유지하려 한다는 것이다. 대기업, 관공서에서 사용되는 기술이 중소 IT 분야에도 적용되었으면 한다.

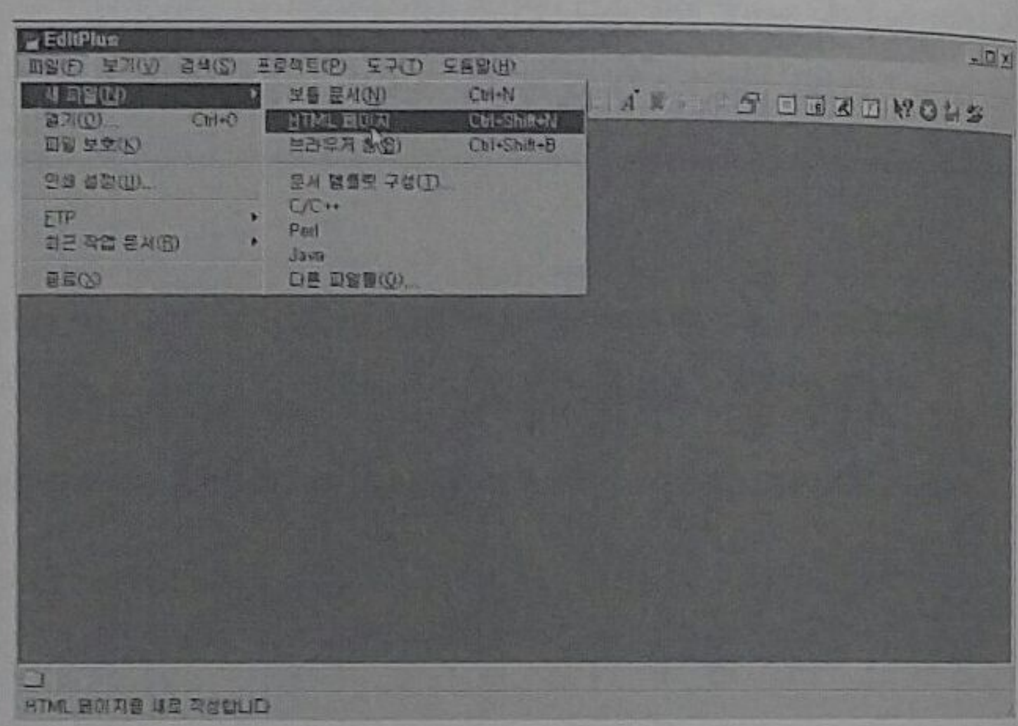
기본 예제

처음 TMJ를 시작하는 독자에게 처음부터 객체화가 무엇이고 어떻게 구현되는지에 대해선 말하지 않겠다. 처음 시작은 재미있어야 하고 따라하기를 통해 익숙해지고 그 익숙함을 통해 본인이 생각한 부분을 구현하는 것이 바람직하다.

입력 값 받기

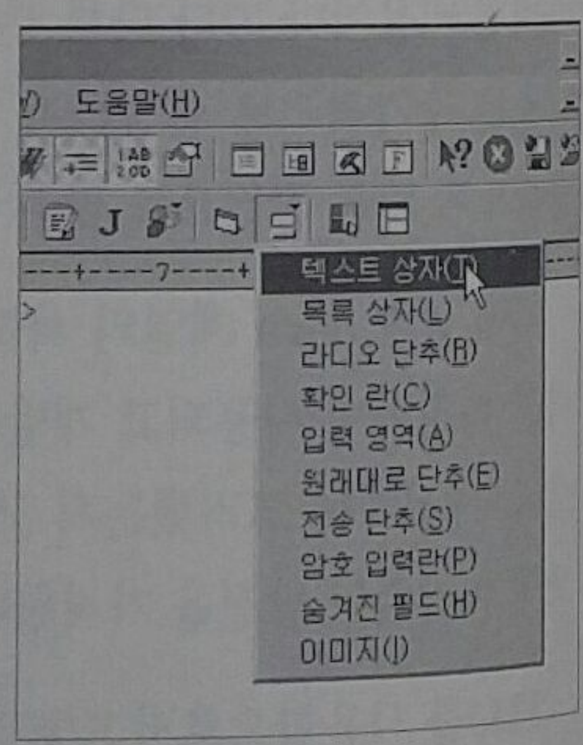
따라하기 입력값 받기

❶ 에디터 툴을 새 파일 → HTML 페이지를 선택한다.



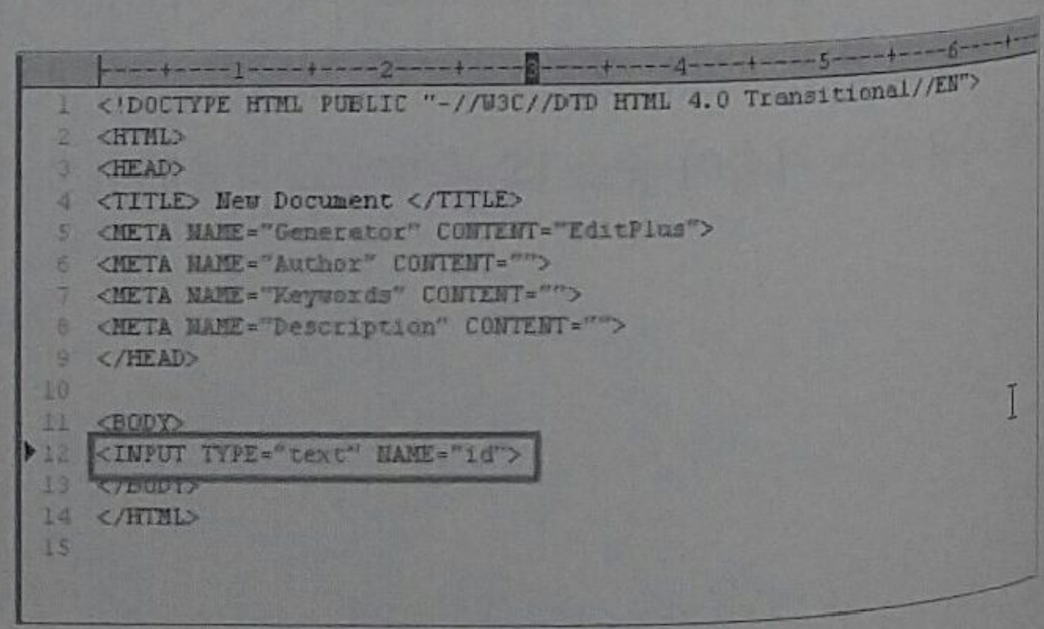
▶ Html 새 파일

❷ 입력도구를 이용하여 텍스트 상자를 선택한다.



▶ 텍스트 상자 선택

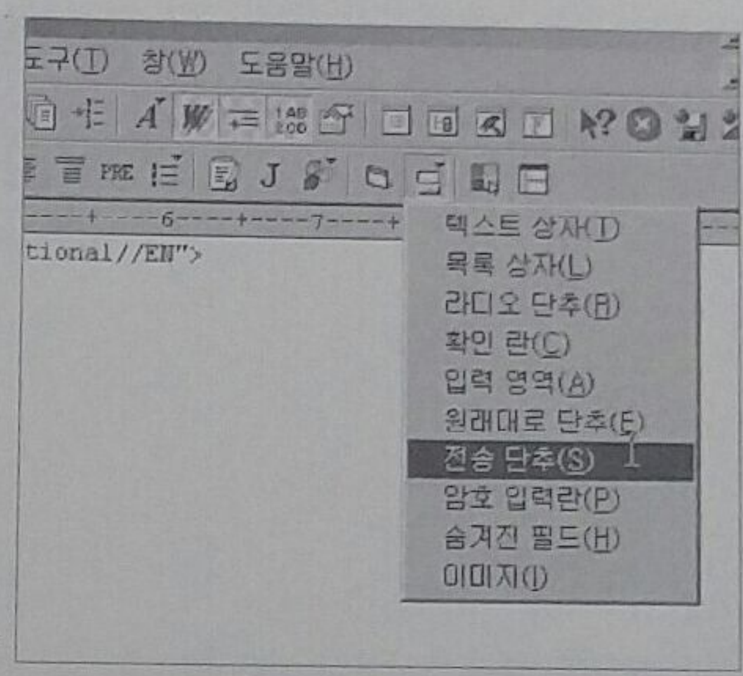
❸ 생성된 텍스트 상자의 이름을 부여한다.



```
1  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
2  <HTML>
3  <HEAD>
4  <TITLE> New Document </TITLE>
5  <META NAME="Generator" CONTENT="EditPlus">
6  <META NAME="Author" CONTENT="">
7  <META NAME="Keywords" CONTENT="">
8  <META NAME="Description" CONTENT="">
9  </HEAD>
10
11 <BODY>
12 <INPUT TYPE="text" NAME="id">
13 </BODY>
14 </HTML>
15
```

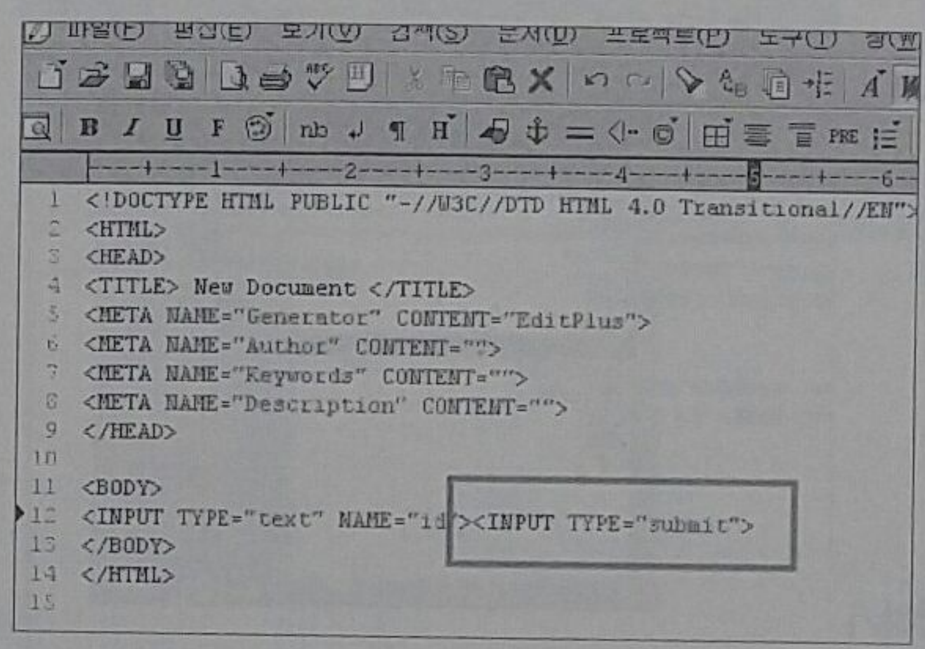
▶ 텍스트 상자의 이름부여

❹ 입력도구 중 전송단추를 선택한다.

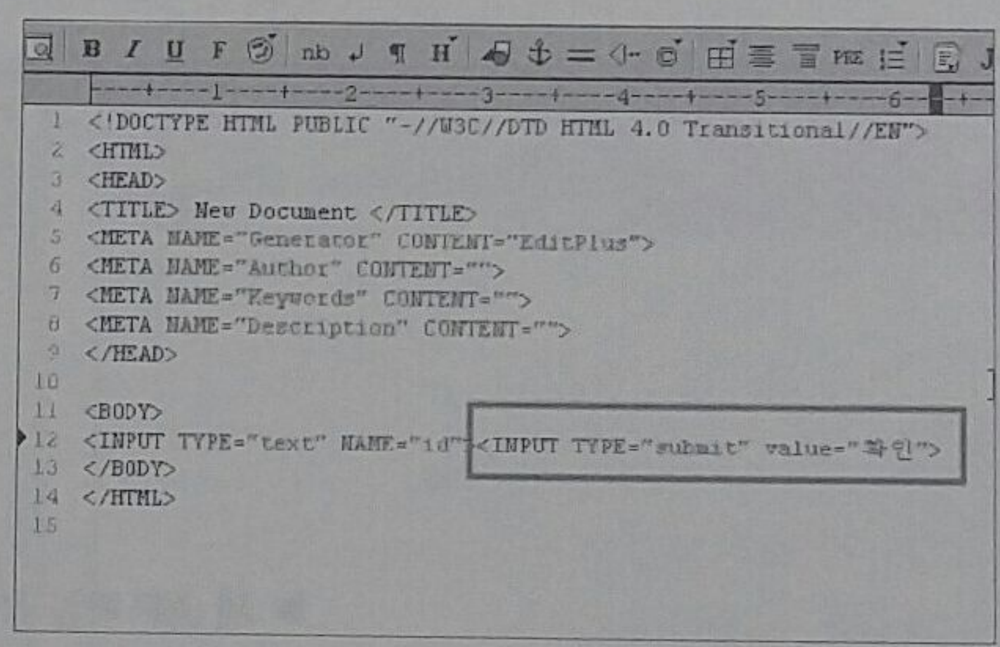


▶ 전송단추 메뉴

❺ 전송단추가 생성이 되면 전송단추의 value값을 입력한다.

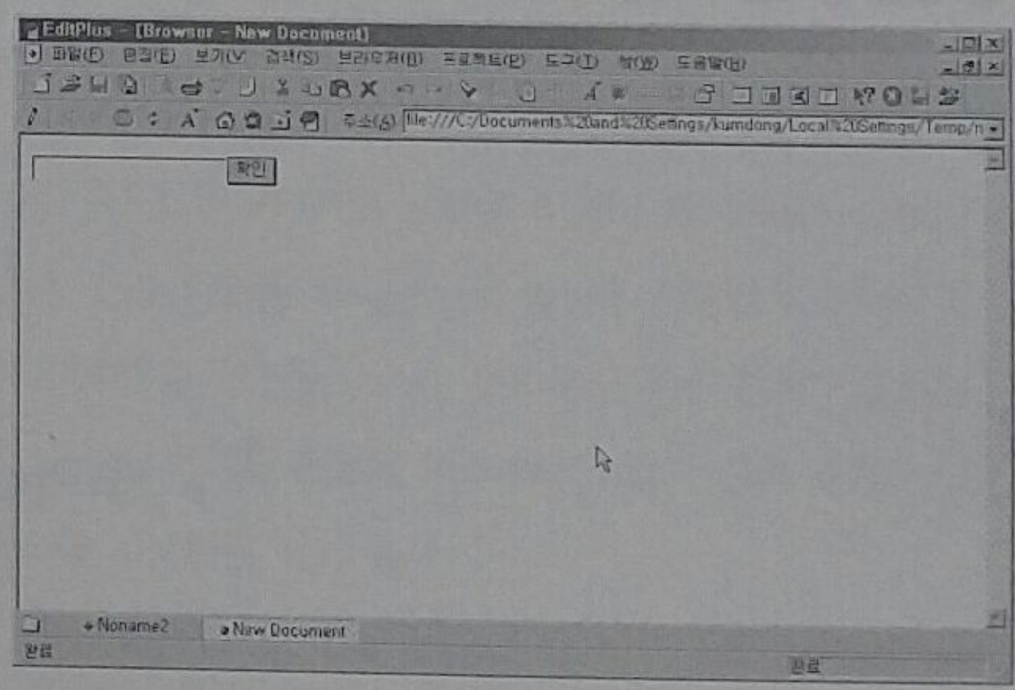


▶ 전송단추 생성 소스



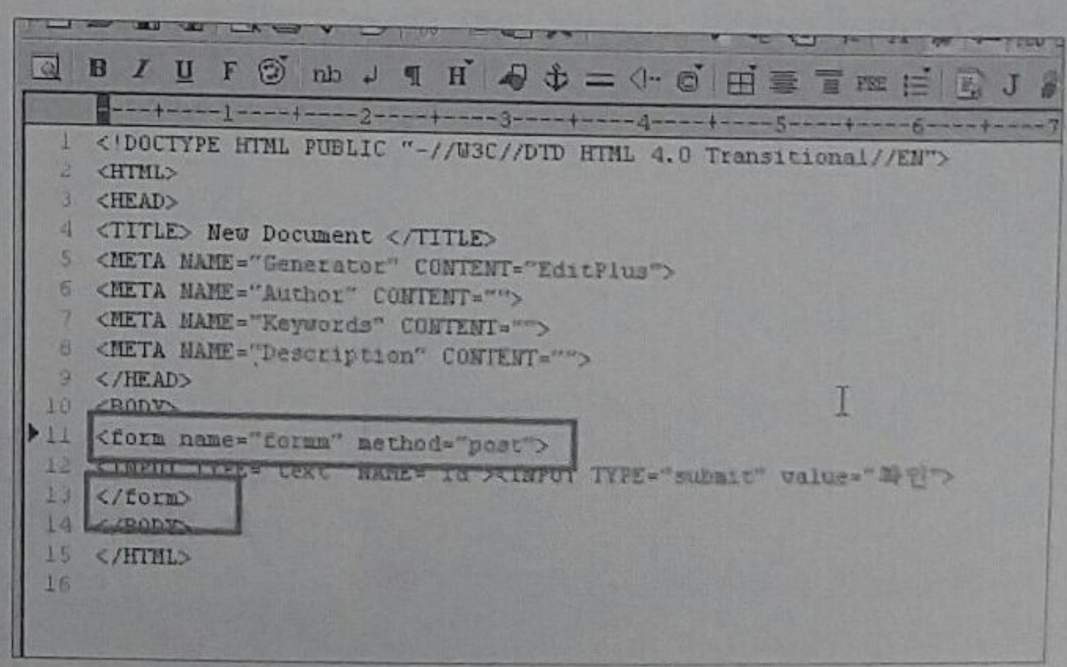
▶ value값 입력

❻ 에디터 툴에서 Ctrl+b 버튼을 동시에 선택하면 미리보기 기능이 있다. 미리보기에서 다시 Ctrl+b 버튼을 동시에 선택하면 원래 소스 작성기로 돌아간다.



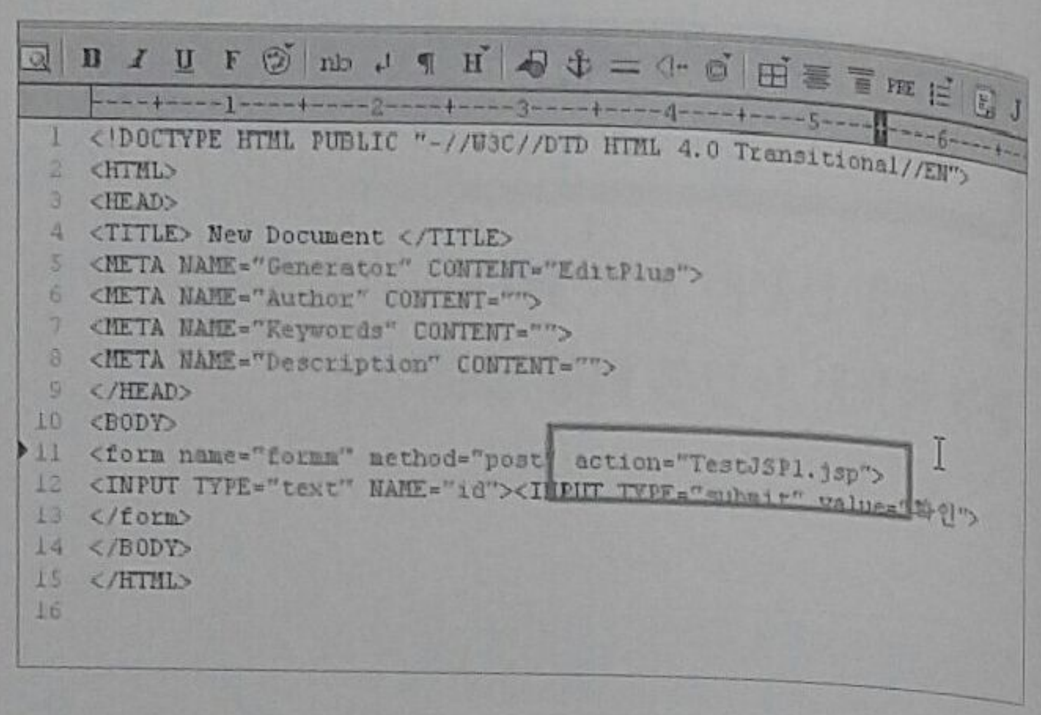
▶ 미리보기

❼ 입력되는 전송 값의 form 방식을 작성한다.



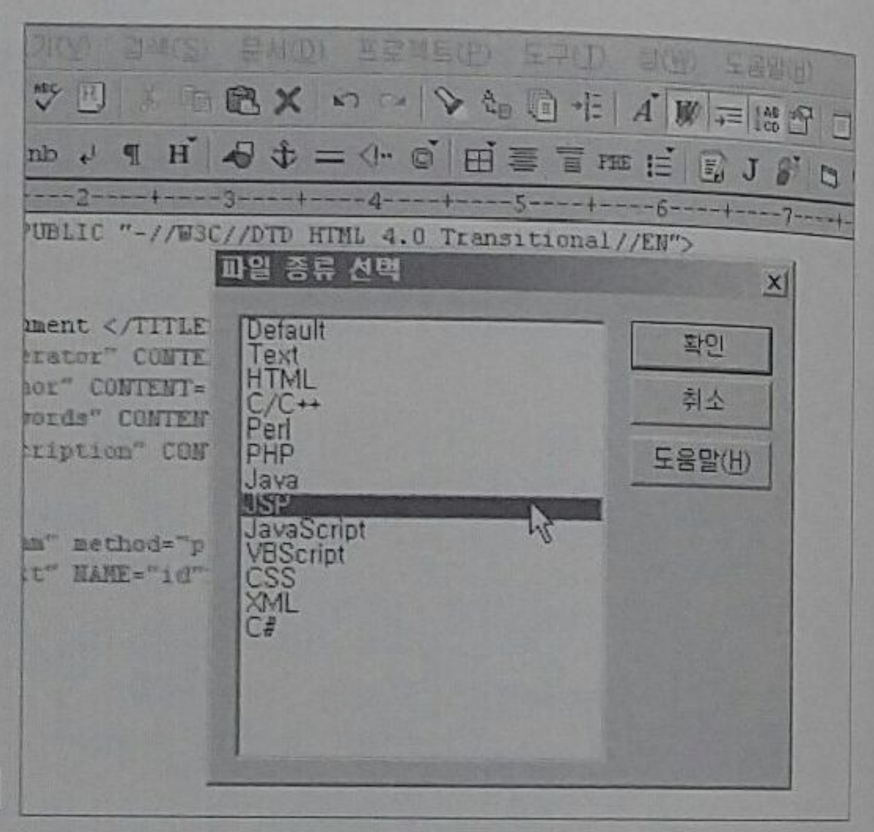
▶ form 방식 작성

❽ form을 지정했다면 해당 form에서 전달받아 action 이 취해질 JSP 파일을 지정한다.



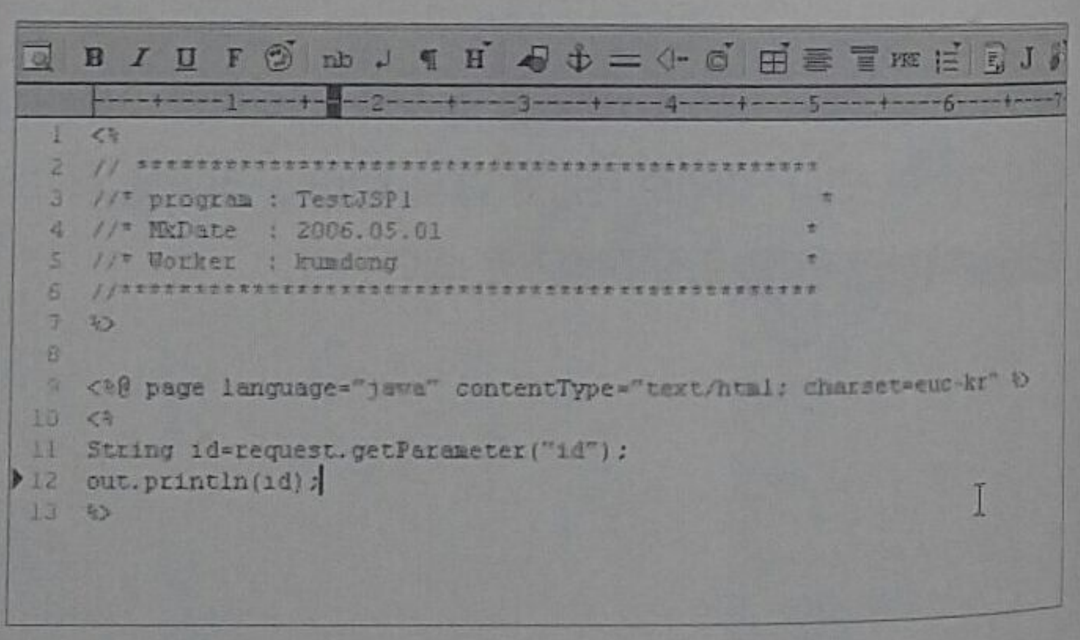
▶ action 설정

❾ 에디터 툴에서 새 파일 → 다른 파일 → JSP를 선택한다.



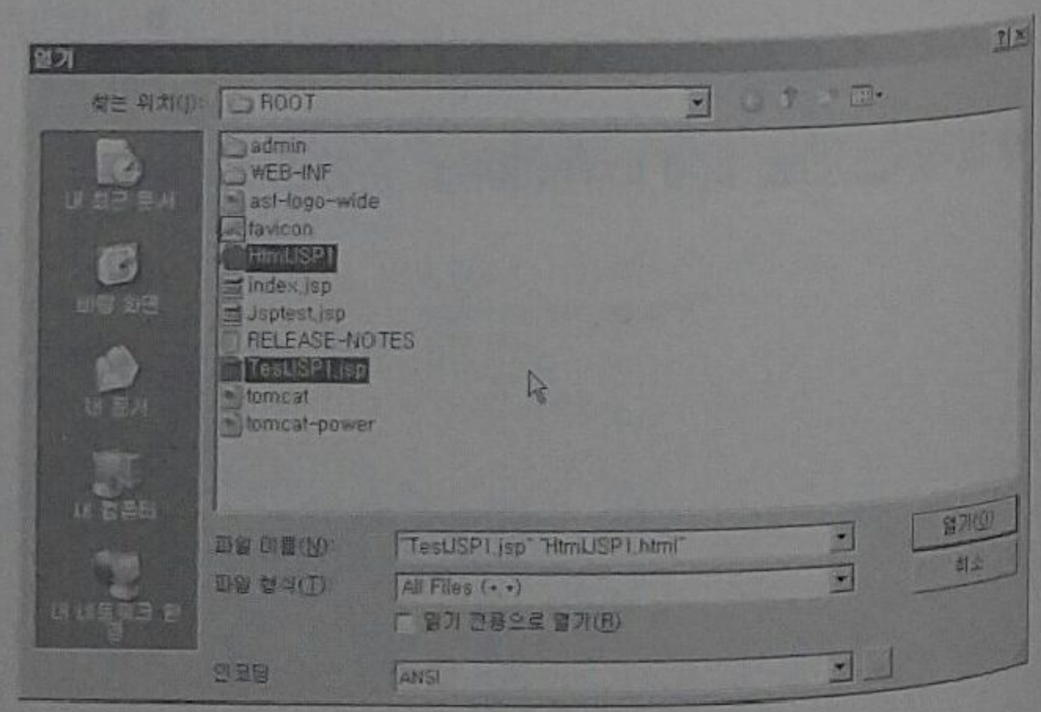
▶ 새 JSP 파일 작성하기

❿ 값을 입력받아 화면에 뿌려주는 소스를 작성한다.



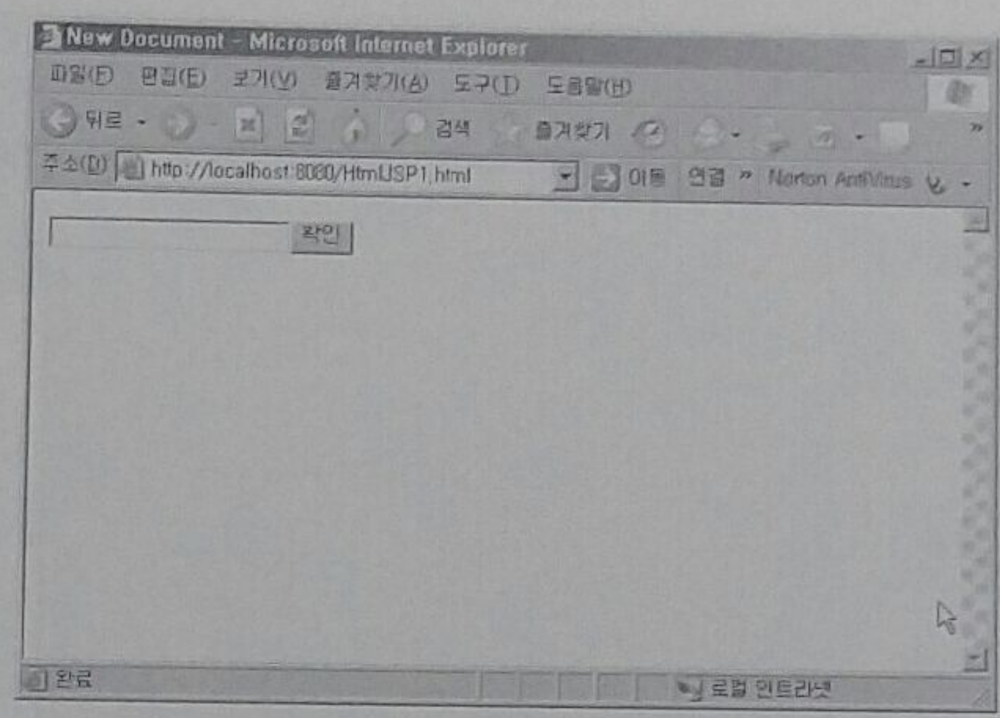
▶ JSP 작성

⓫ 값을 보낼 Html과 JSP를 HtmlJSP1.html과 TestJSP1.jsp으로 Tomcat의 기본 Root에 저장한다.



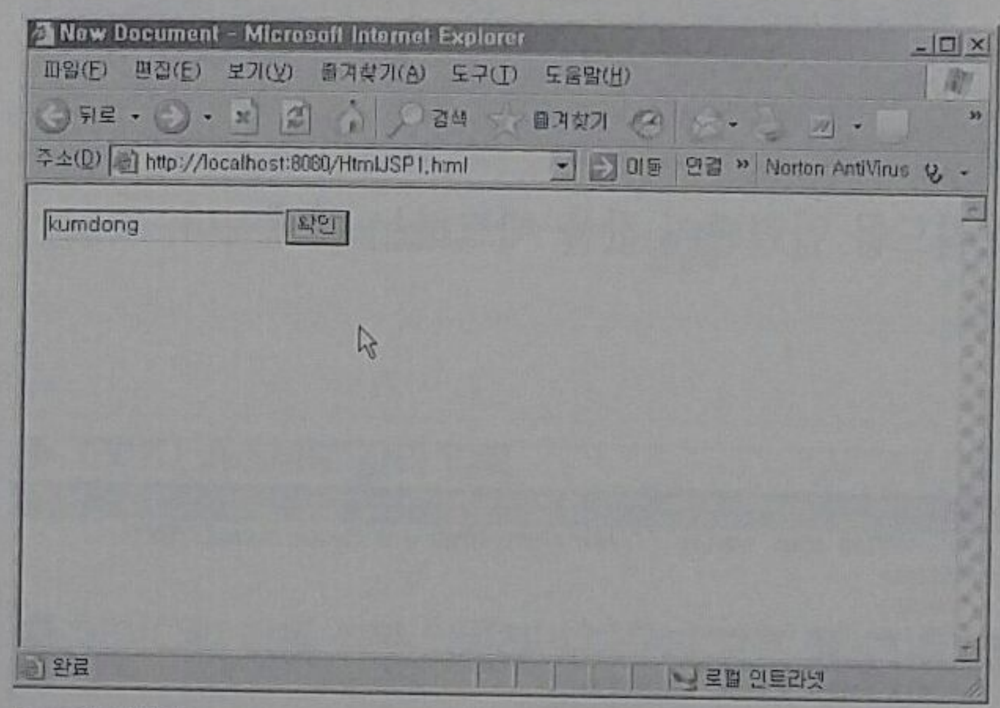
▶ 파일의 저장

⑫ 브라우저에서 HtmlJSP1.html을 호출한다.

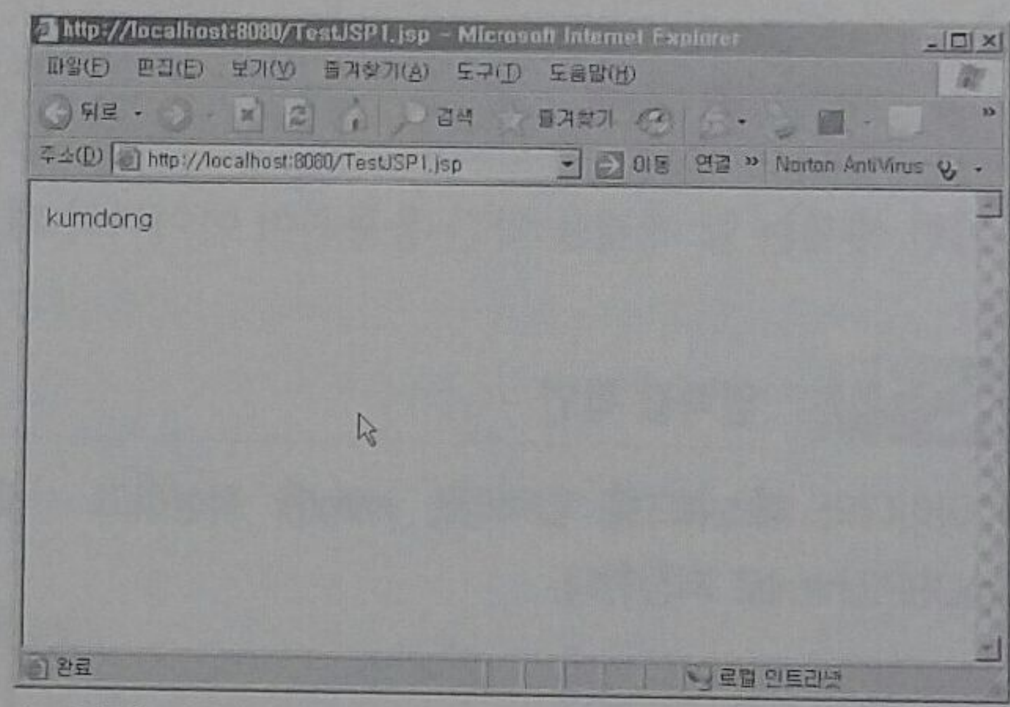


▶ 브라우저에서의 호출

⑬ 값을 입력하고 확인 버튼을 클릭하면 입력된 값이 화면에 출력이 된다.



▶ 값 입력



▶ 입력 값 결과 확인

TIP

Get 방식과 Post 방식

값을 전달하는 method의 방식에 따라 한글 입력 시 결과 값이 깨진다면 2가지의 조치를 취해야 할 것이다. 그러기 전에 먼저 Get과 Post 방식의 차이에 대해서 간단히 알아보자. 쉽게 차별을 두자면 get 방식은 전달할 파라메타의 값이 브라우저 주소 창에 보이기 때문에 정보유출의 가능성이 다분하다. 반면에 post는 브라우저 주소 창에 전달되는 파라메타의 값이 보이지 않기 때문에 안정성을 보장할 수 있으며 get 방식은 전달할 파라메타의 값의 길이 제한이 있지만 post는 제한이 없다. 특별한 경우를 제외하고는 post 방식을 사용하는 것이 좋다.

– Get 방식

get 방식으로 파라메타를 보내는 방식이라면 tomcat5의server.xml 파일을 열어 추가되는 부분이 있다. server.xml 파일에서 URIEncoding="KSC5601"을 추가함으로서 get 방식의 한글 파라메타 깨짐 현상을 방지한다.

```
        IP address of the remote client.
    -->

    <!-- Define a non-SSL HTTP/1.1 Connector on port 8080 -->
    <Connector
port="8080"                maxHttpHeaderSize="8192"
               maxThreads="150" minSpareThreads="25" maxSpareThreads="75"
               enableLookups="false" redirectPort="8443" acceptCount="100"
               connectionTimeout="20000" disableUploadTimeout="true" URIEncoding="KSC5601"/>
    <!-- Note : To disable connection timeouts, set connectionTimeout value
     to 0 -->

    <!-- Note : To use gzip compression you could set the following properties :
```

▶ tomcat5의 server.xml 파일 수정

다음 페이지에 계속 ➲

T/P

– Post 방식

post로 값을 전달받았다면 소스 상에서 request.set CharacterEncoding("euc-kr");으로 처리하면 한글의 깨짐을 방지한다.

```
<%
// ***********************************************
//* program : TestJSP1                          *
//* MkDate  : 2006.05.01                        *
//* Worker  : kumdong                           *
//***********************************************
%>

<%@ page language="java" contentType="text/html; charset=euc-kr" %>
<%
request.setCharacterEncoding("euc-kr");
String id=request.getParameter("id");
out.println(id);
%>
```

▶ request.setCharacterEncoding

입력 값 체크

이번 예제는 앞 예제를 약간 응용하여 아이디와 패스워드를 입력받아 값을 체크하는 예제이다.

따라하기 입력값 확인

❶ 아이디와 패스워드를 입력하는 Html을 작성하고 HtmlJSP2.html로 저장한다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<META NAME="Generator" CONTENT="EditPlus">
<META NAME="Author" CONTENT="">
<META NAME="Keywords" CONTENT="">
<META NAME="Description" CONTENT="">
</HEAD>
<BODY>
<form name="form" method="post" action="TestJSP2.jsp">
<INPUT TYPE="text" NAME="id">
<INPUT TYPE="password" NAME="pw">
<INPUT TYPE="submit" value="확인">
</form>
</BODY>
</HTML>
```

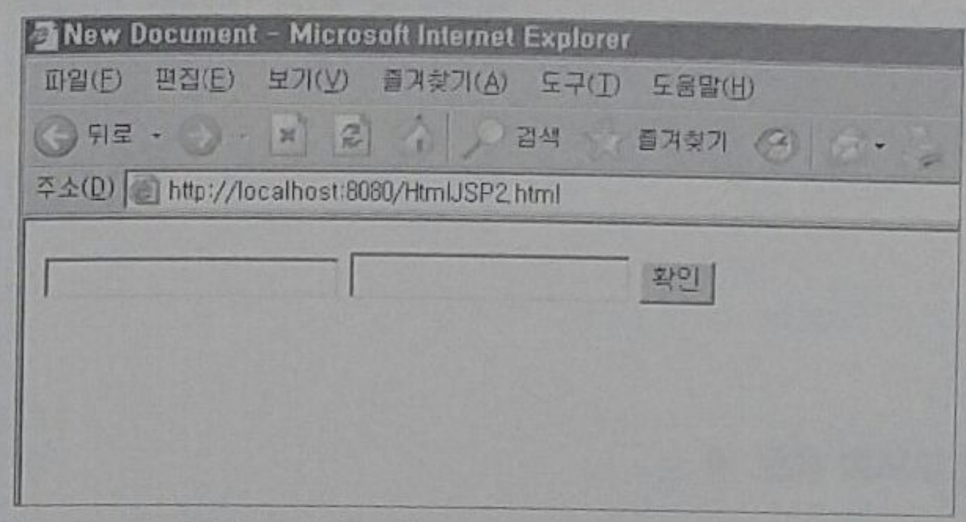
▶ HtmlJSP2.html의 작성

❷ 아이디, 패스워드의 값을 받아 체크하는 JSP를 작성하고 TestJSP2.jsp로 저장한다.

```
<%
// ***********************************************
//* program : TestJSP1                          *
//* MkDate  : 2006.05.01                        *
//* Worker  : kumdong                           *
//***********************************************
%>

<%@ page language="java" contentType="text/html; charset=euc-kr" %>
<%
request.setCharacterEncoding("euc-kr");
String id=request.getParameter("id");
String pw=request.getParameter("pw");

if(!"kumdong".equals(id))
{
    out.println("아이디가 잘못되었습니다.");
}

if(!"kumdong".equals(pw))
{
    out.println("패스워드가 잘못되었습니다.");
}

if( "kumdong".equals(id) && "kumdong".equals(pw))
{
    out.println(id+"님 화이투~~~^);
}


%>
```

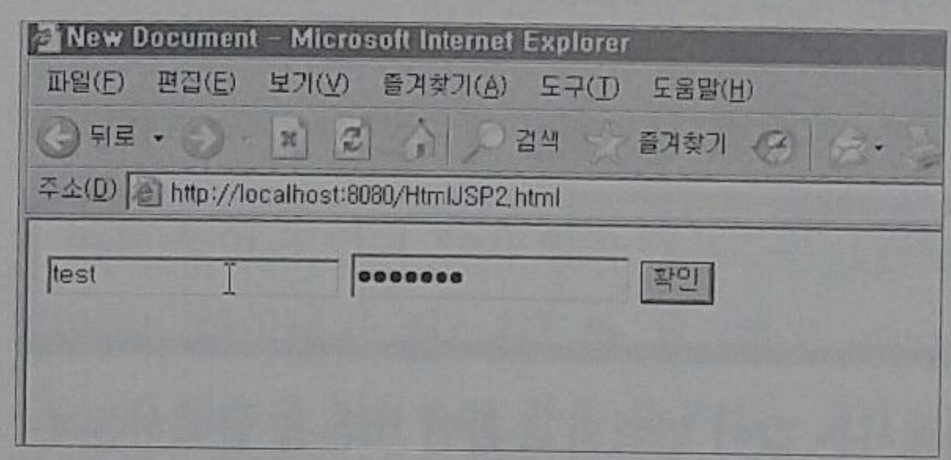
▶ TestJSP2.jsp의 작성

❸ 브라우저를 열고 HtmlJSP2.html을 호출한다.



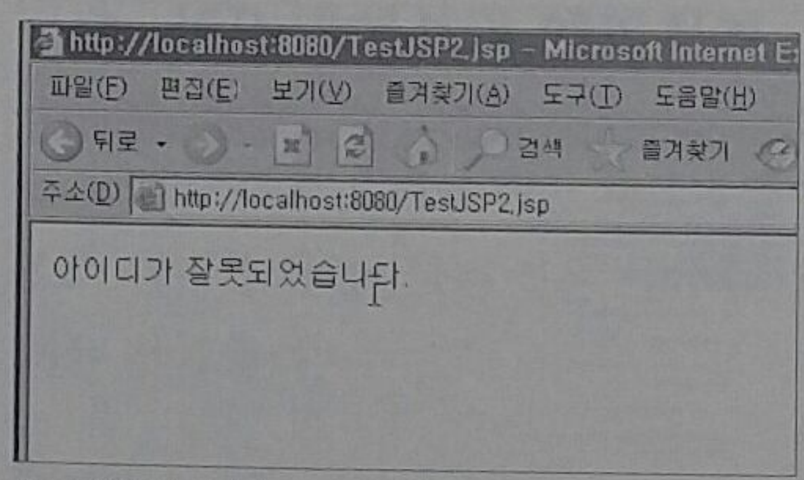
▶ HtmlJSP2.html의 호출

❹ 잘못된 아이디, 패스워드 값을 입력하고 확인 버튼을 클릭한다.



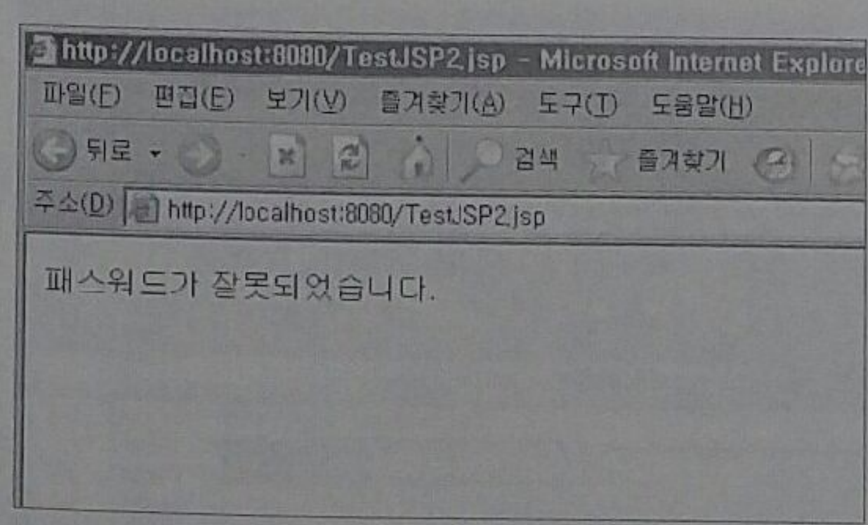
▶ 아이디 패스워드 값의 입력

❺ 아이디의 값을 잘못 입력하면 다음과 같은 메시지를 출력한다.



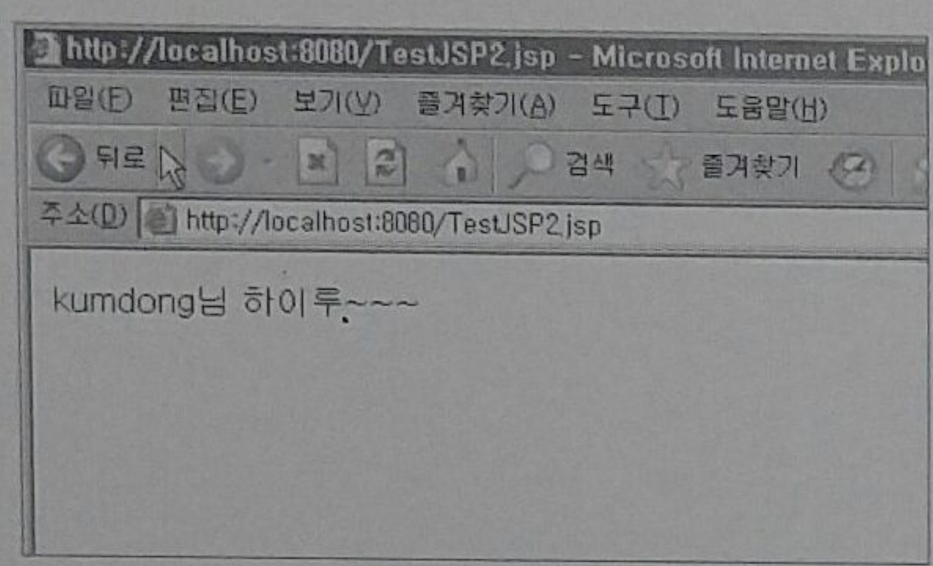
▶ 아이디 오류

❻ 잘못된 패스워드를 입력하고 확인 버튼을 클릭한다.



▶ 패스워드 오류

❼ 정확한 아이디와 패스워드를 입력하고 확인 버튼을 클릭한다.



▶ 성공 메시지

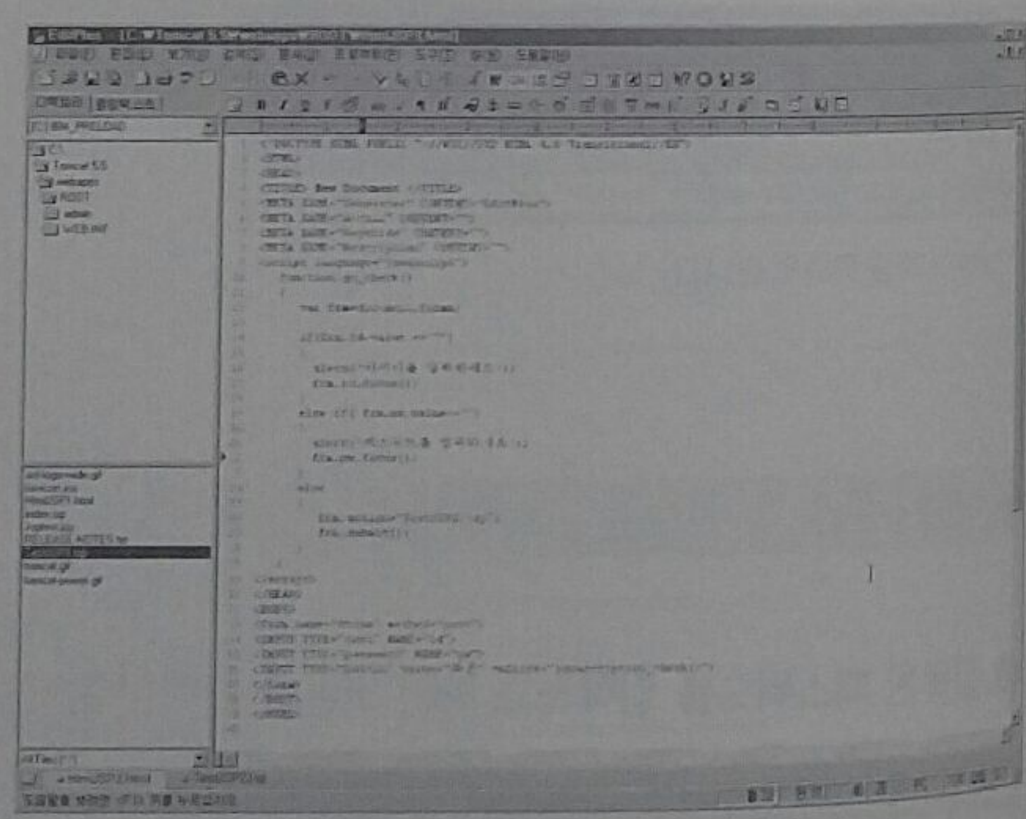
이번 예제를 통해 두개의 입력 값을 받아 들여 어떻게 체크하는지 살펴보았다. 나중에 데이터베이스와 연계하는 공부를 하면 저장된 데이터를 불러와서 체크하는 로그인 처리가 된다.

스크립트를 이용한 체크

넘겨지는 파라메타 값을 필수적으로 입력하게 하기 위해서는 값이 입력되고 확인 버튼을 클릭하는 순간 체크한다. 이러한 방법을 택하는 것은 서버의 많은 부하를 줄이기 위해서 이다.

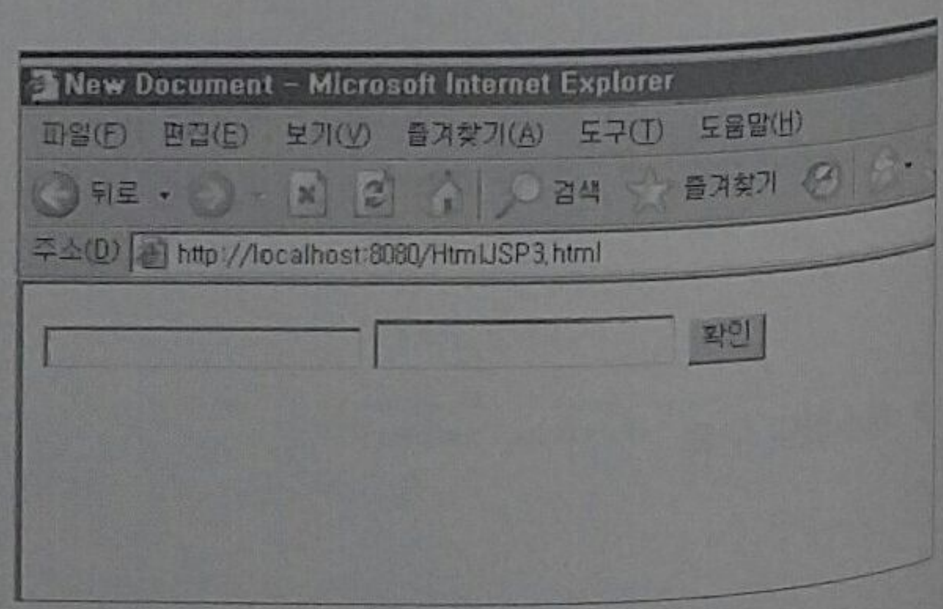
따라하기 스크립트를 이용한 체크

❶ 앞 예제에서 만들어진 HtmlJSP2.html 파일을 불러와서 스크립트를 작성하고 HtmlJSP3.html로 저장한다.



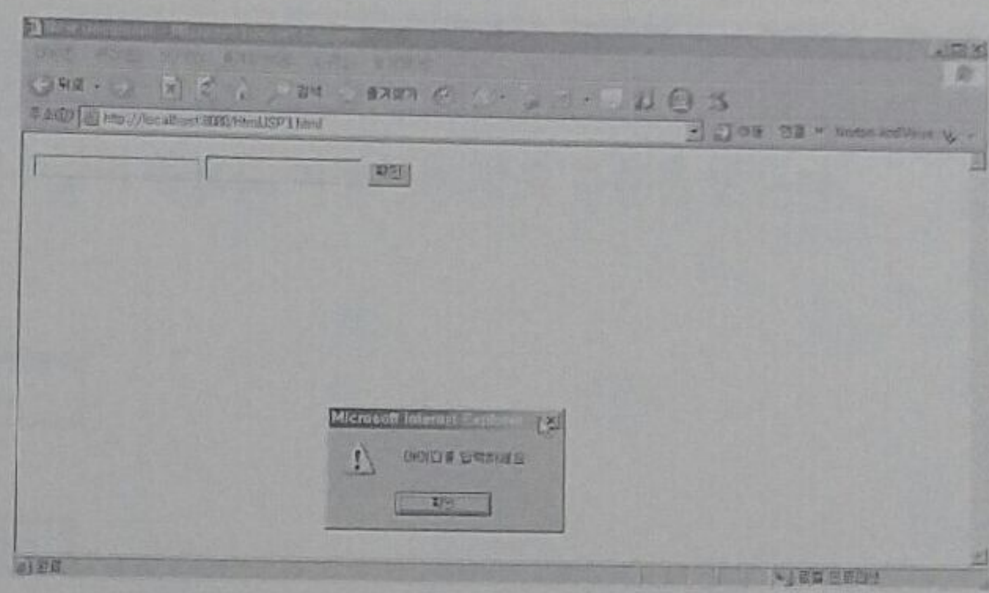
▶ HtmlJSP3.html 작성

❷ 브라우저에서 HtmlJSP3.html을 호출한다.



▶ HtmlJSP3.html 호출

❸ 아이디 또는 패스워드 중 입력값 없이 확인 버튼을 클릭하면 매세지 창을 출력한다.



▶ 오류 메시지 출력

이번 예제에서는 필수적으로 입력되는 값을 스크립트로 처리해 보았다. 확인 버튼을 클릭하였을 때 onClick()이라는 스크립트 함수를 호출하고 호출된 onClick() 함수는 go_check()라는 function을 찾는다. var frm = document.formm; 이라고 선언한 부분은 폼에 적용되어 있는 오브젝트에 대한 사용을 편리하기 위해서 변수로 만들어진 것이다.

예를 들어 선언이 되어 있지 않는다면 document.formm.id.value라고 코딩의 시간이 좀 더 길어질 것이다. 아이디와 패스워드를 체크하고 alert(" ");를 통해 사용자에게 메시지를 출력한다. 그리고 focus() 함수를 이용하여 값을 요구하는 오브젝트에 커서를 이동하게 시킨다. 더 많은 스크립트를 배우고자 한다면 웹을 통해 자료를 얻거나 따로 공부하는 것이 좋다.

JSP 내에서의 에러처리

스크립트에서 입력에 대한 체크 부분이 있다면 JSP 소스 내에서 에러를 처리할 수 있다. 하지만 JSP 소스 내에서 처리한다는 것은 서버에서의 처리이기 때문에 입력체크에 대한 부분은 스크립트에서 처리하는 것이 좋다.

따라하기 에러처리

❶ HtmlJSP2.html을 이용하여 HtmlJSP4.html으로 저장한다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<META NAME="Generator" CONTENT="EditPlus">
<META NAME="Author" CONTENT="">
<META NAME="Keywords" CONTENT="">
<META NAME="Description" CONTENT="">
</HEAD>
<BODY>
<form name="formm" method="post" action="TestJSP3.jsp">
<INPUT TYPE="text" NAME="id">
<INPUT TYPE="password" NAME="pw">
<INPUT TYPE="submit" value="확인">
</form>
</BODY>
</HTML>
```

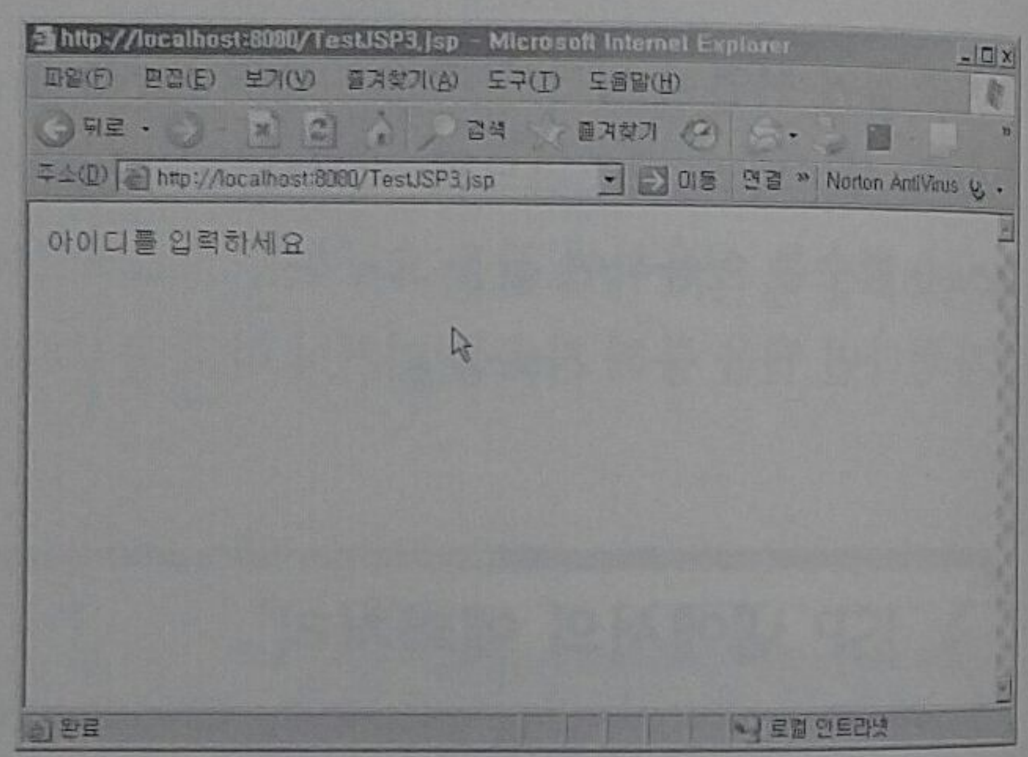
▶ HtmlJSP4.html

❷ TestJSP3.jsp 파일을 작성한다.

```jsp
<%
// ***********************************************
//* program : TestJSP1                            *
//* MkDate  : 2006.05.01                          *
//* Worker  : kumdong                             *
//***********************************************
%>

<%@ page language="java" contentType="text/html; charset=euc-kr" %>
<%
request.setCharacterEncoding("euc-kr");
String id=request.getParameter("id");
String pw=request.getParameter("pw");



if("".equals(id) )
{
    out.println("아이디를 입력하세요");
}

if( !"".equals(id) && "".equals(pw) )
{
    out.println("암호를 입력하세요");
}

if( !"".equals(id) && !"".equals(pw) )
{
    out.println(id+"님 반갑습니다");
}

%>
```

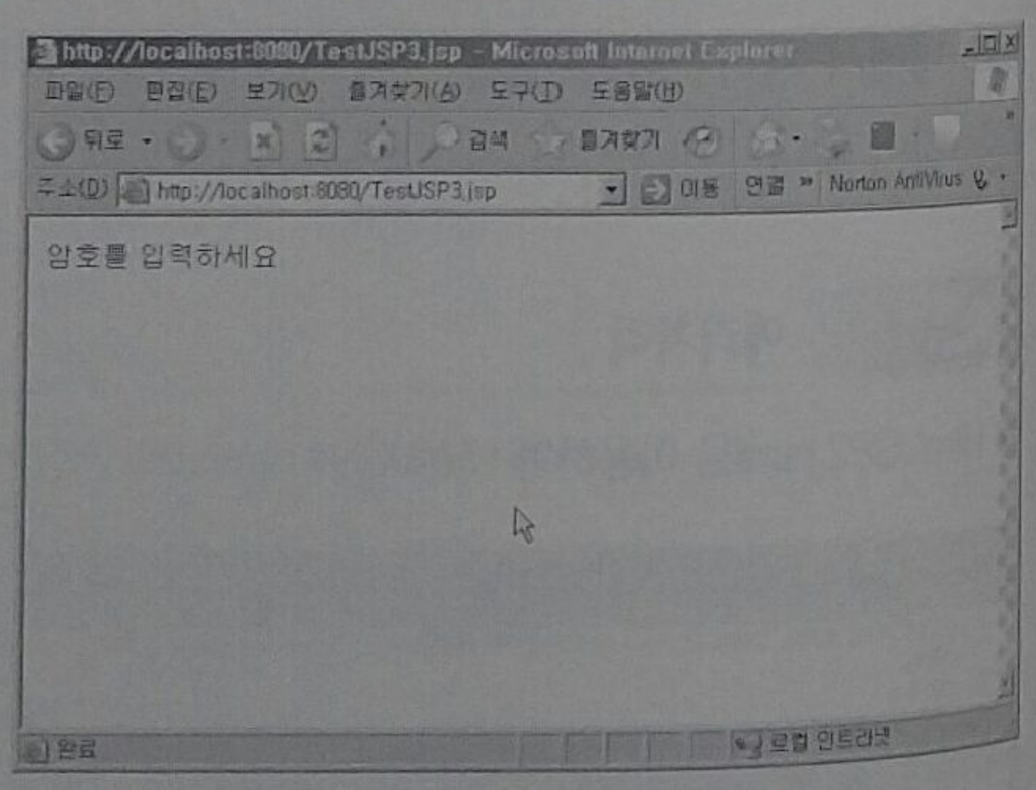
▶ TestJSP3.jsp

❸ 입력값 없이 입력을 하면 다음과 같은 메시지를 출력한다.



▶ 에러처리

❹ 아이디만 입력을 하고 암호 미입력 시 다음과 같은 메시지를 출력한다.



▶ 암호입력 에러 메시지

간단한 JSP에서의 에러처리를 해 보았다. 하지만 뭔가가 빠진 듯한, 부족한 느낌이 당연히 들 것이다. 입력오류 처리를 하고 나서 메시지만 출력한 후 입력 홈으로 이동을 해야 사용자가 다시 입력을 할 수 있도록 해야 정상적이 될 것이다. 그럼 어떠한 방법으로 처리를 해야 할까? 여러 가지 방법이 있지만 보통 3가지의 방법을 사용한다.

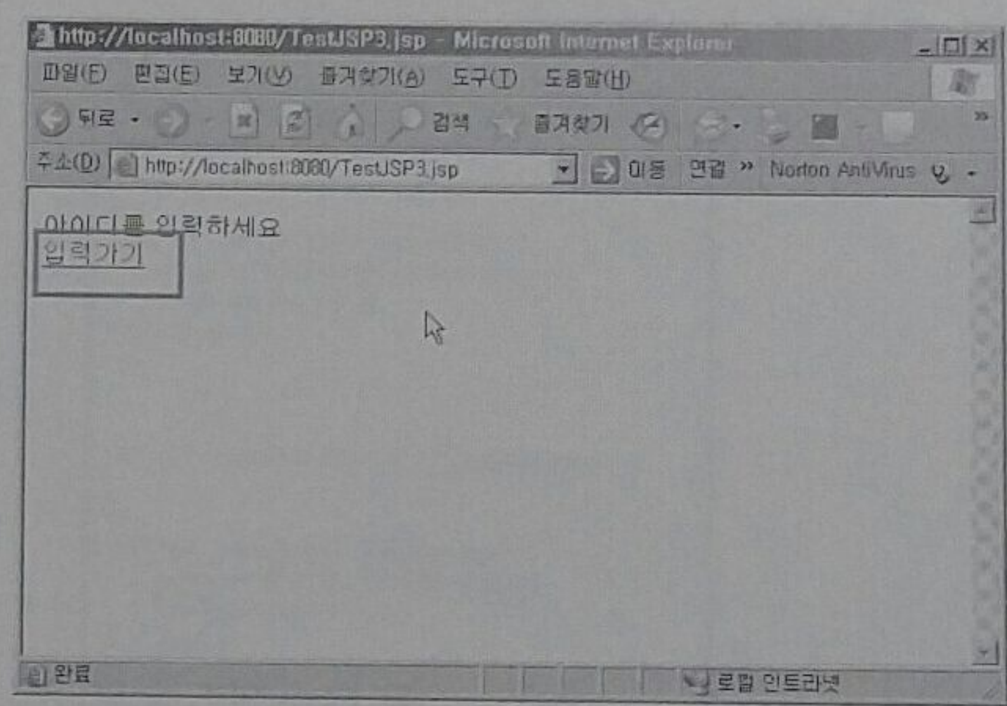
기본적인 방법

에러 메시지를 출력하고 뒤로 가기 버튼을 생성하거나 입력 폼으로 이동하는 링크 부분을 만든다.

```
if("".equals(id) )
{
    out.println("아이디를 입력하세요<br> <a href='HtmlJSP4.html'>입력가기</a>");
}

if( !"".equals(id) && "".equals(pw) )
{
    out.println("암호를 입력하세요 <br> <a href='HtmlJSP4.html'>입력가기</a>");
}

if( !"".equals(id) && !"".equals(pw) )
```

▶ 링크작성

입력값 없이 입력을 하면 에러 메시지를 출력하고 입력가기를 클릭하면 입력 페이지로 이동한다.



▶ 입력가기 링크

메시지 처리방법

변수 값을 주어 에러를 처리하는 방법으로 JSP 소스 내에 스크립트를 이용하고 에러 메시지의 값을 입력 페이지에 파라메타로 넘겨주어 onLoad()를 이용하여 메시지를 출력한다. 단 입력 페이지가 Html이 아니라 jsp가 되어야 하는 단점이 있다. 먼저 HtmlJSP4.html을 HtmlJSP5.jsp으로 저장하고 수정하거나 새로이 작성한다.

```
<%@ page language="java" contentType="text/html; charset=euc-kr" %>
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<META NAME="Generator" CONTENT="EditPlus">
<META NAME="Author" CONTENT="">
<META NAME="Keywords" CONTENT="">
<META NAME="Description" CONTENT="">
</HEAD>
<BODY>
<form name="form" method="post" action="TestJSP3.jsp">
<INPUT TYPE="text" NAME="id">
<INPUT TYPE="password" NAME="pw">
<INPUT TYPE="submit" value="확인">
</form>
</BODY>
</HTML>
```

▶ HtmlJSP5.jsp의 작성

TestJSP5.jsp을 작성한다. 메시지 값을 처리하기 위해 int message를 선언하고 메시지의 값을 전달하기 위해 스크립트로 처리하여 입력 페이지에 전달한다.

```
12  String id=request.getParameter("id");
13  String pw=request.getParameter("pw");
14
15  //메세지 처리를 구분하기 위한 변수선언
16  int message=0;
17
18
19  if("".equals(id) )
20  {
21    message=1;
22  }
```

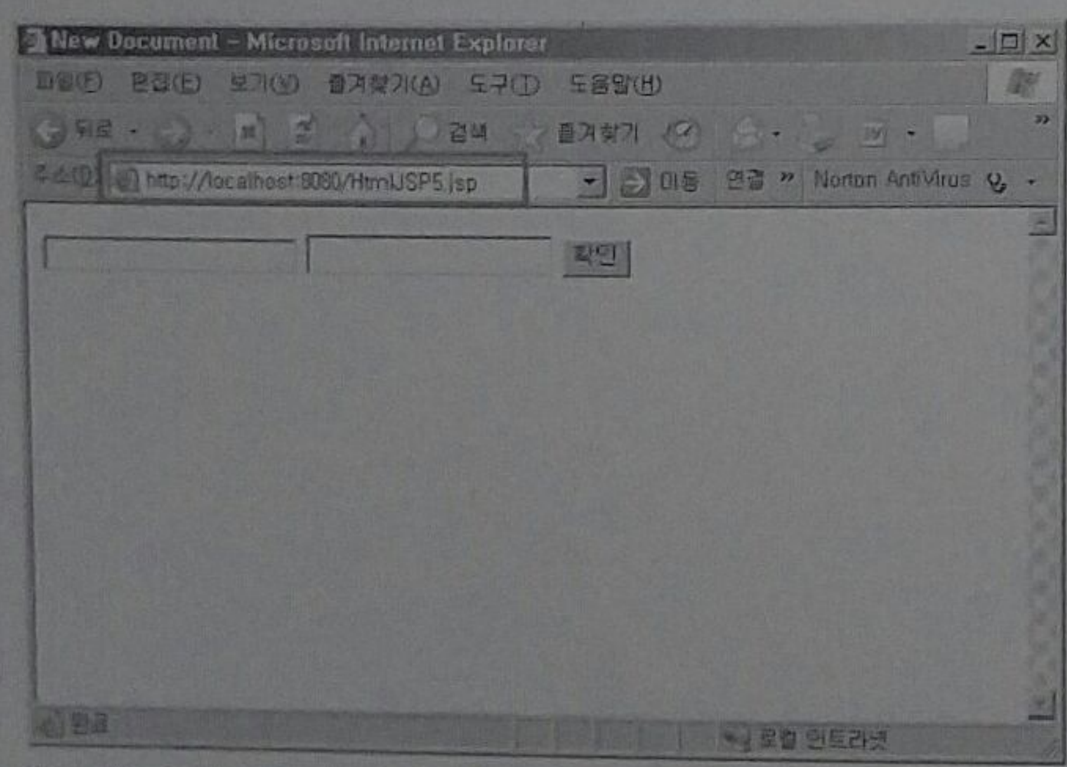
▶ TestJSP5.jsp

다시 HtmlHSP5.jsp 파일을 열어 TestJSP5.jsp에서 전달받아지는 msg값을 처리한다. 이때 자바스크립트의 onLoad() 함수를 사용하여 에러 메시지를 출력한다.

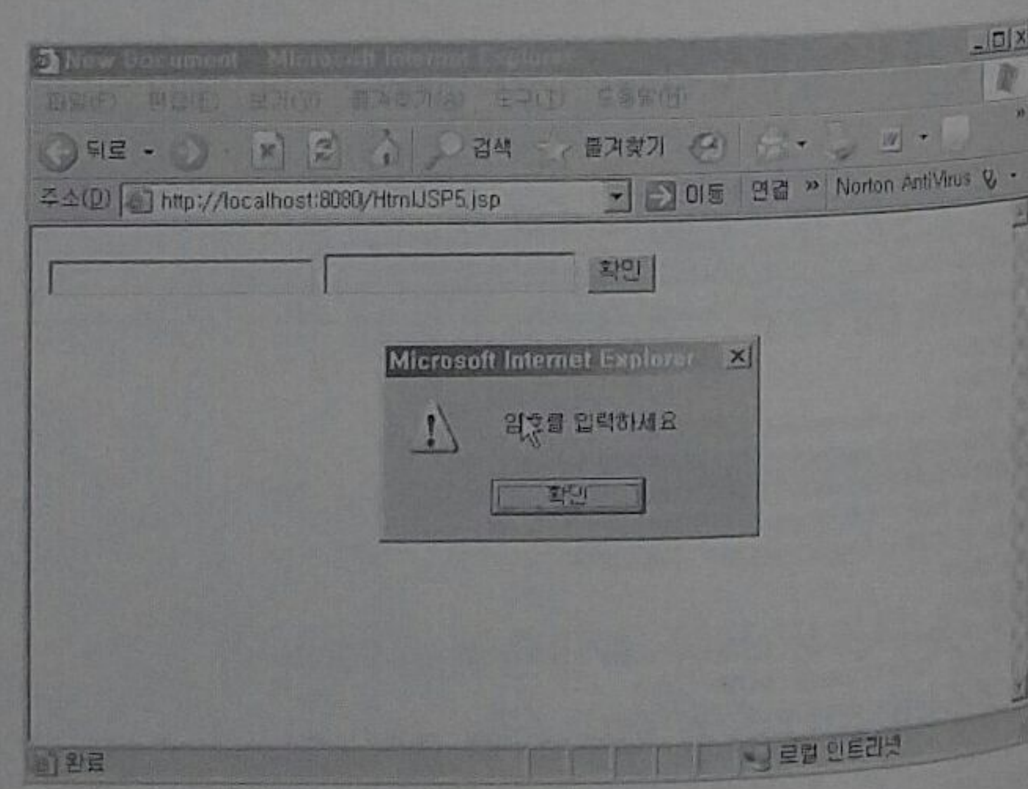
```
1  <%@ page language="java" contentType="text/html; charset=euc-kr" %>
2  <%
3  String msg=request.getParameter("msg");
4  %>
5  <!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
6  <HTML>
7  <HEAD>
8  <TITLE> New Document </TITLE>
9  <META NAME="Generator" CONTENT="EditPlus">
10 <META NAME="Author" CONTENT="">
11 <META NAME="Keywords" CONTENT="">
12 <META NAME="Description" CONTENT="">
13 <script language="javascript">
14       function show_msg(msg)
15       {
16           if(msg ==1 )
17           {
18               alert('아이디를 입력하세요');
19               document.formm.id.focus();
20           }
21           else if(msg==2)
22           {
23               alert('암호를 입력하세요');
24               document.formm.pw.focus();
25           }
26       }
27 </script>
28 </HEAD>
29 <BODY onLoad="show_msg('<%=msg%>');">
30 <form name="formm" method="post" action="TestJSP5.jsp">
31 <INPUT TYPE="text" NAME="id">
32 <INPUT TYPE="password" NAME="pw">
```

▶ HtmlJSP5.jsp의 수정

즉 순서를 설명하자면 입력페이지에서 입력 값을 체크하여 입력 값이 없다면 message 변수 값에 따라 hidden으로 숨겨진 msg의 값을 HtmlJSP5.jsp에 submit()을 통하여 전달되고 HtmlJSP5.jsp 파일에서는 이 값을 받아들여 스크립트 함수 onLoad()의 show_msg() function에 msg값을 전달하여 그 값에 따라 alert()를 이용하여 메시지를 처리하고 커서의 이동을 해당 오브젝트에 이동시킨다.



▶ HtmlJSP5.jsp이 호출



▶ 암호 미 입력 시 에러 메시지

스크립트 이용하기

미입력 값이 있을 경우 바로 스크립트로 메시지를 처리하고 입력 페이지로 이동한다.

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.0 Transitional//EN">
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<META NAME="Generator" CONTENT="EditPlus">
<META NAME="Author" CONTENT="">
<META NAME="Keywords" CONTENT="">
<META NAME="Description" CONTENT="">
</HEAD>
<BODY>
<form name="form" method="post" action="TestJSP6.jsp">
<INPUT TYPE="text" NAME="id">
<INPUT TYPE="password" NAME="pw">
<INPUT TYPE="submit" value="확인">
</form>
</BODY>
</HTML>
```

▶ HtmlJSP6.html 작성

TestJSP6.jsp에서 미 입력받는 값의 에러처리를 스크립트로 처리한다.

```
if("".equals(id) )
{
%>
<script language="javascript">
    alert('아이디를 입력하세요');
    history.go(-1);
</script>
<%
}

if( !"".equals(id) && "".equals(pw) )
{
%>
<script language="javascript">
    alert('암호를 입력하세요');
    history.go(-1);
</script>
<%
}

if( !"".equals(id) && !"".equals(pw) )
{
```

▶ TestJSP6.jsp

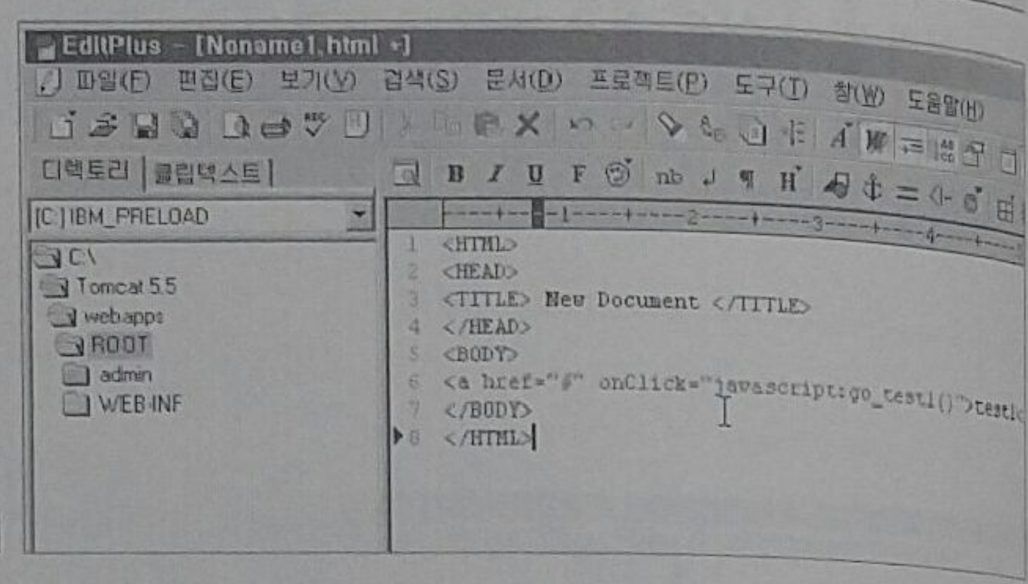
TestJsp6.jsp을 보면 에러 발생 시 바로 스크립트 구문을 사용하고 스크립트 함수인 history.go(-1)을 이용하여 메시지를 출력하고 확인 버튼을 클릭하면 입력 페이지로 이동한다. 그러나 화면상에 에러 메시지가 출력이 되면 브라우저의 화면이 보기가 민망하다는 단점이 있다. 또한 기타 에러 페이지를 만들어 처리하는 방법도 있다.

기본적인 스크립트 함수

프로그램 작업을 하다보면 스크립트의 필요성을 많이 느끼게 된다. 프로그램에서는 특정한 몇 가지의 스크립트를 알고 있으면 나머지 부분은 필요할 때마다 그때그때 찾아서 사용해도 된다. 그럼 몇 가지의 스크립트 부분에 대해서 살펴보도록 하자.

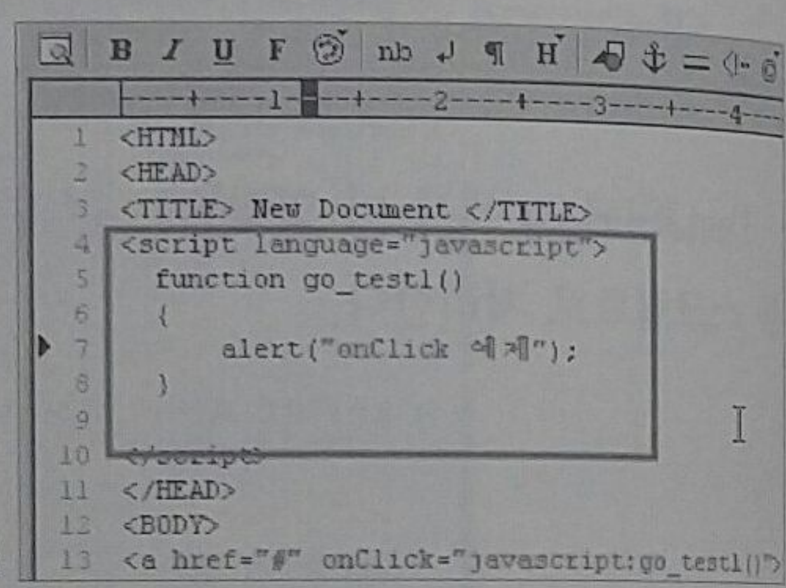
onClick()

onClick() 스크립트 함수는 많이 사용되는데 버튼이나 이미지 또는 글자 등에 클릭했을 경우의 이벤트를 발생시키는 함수이다.



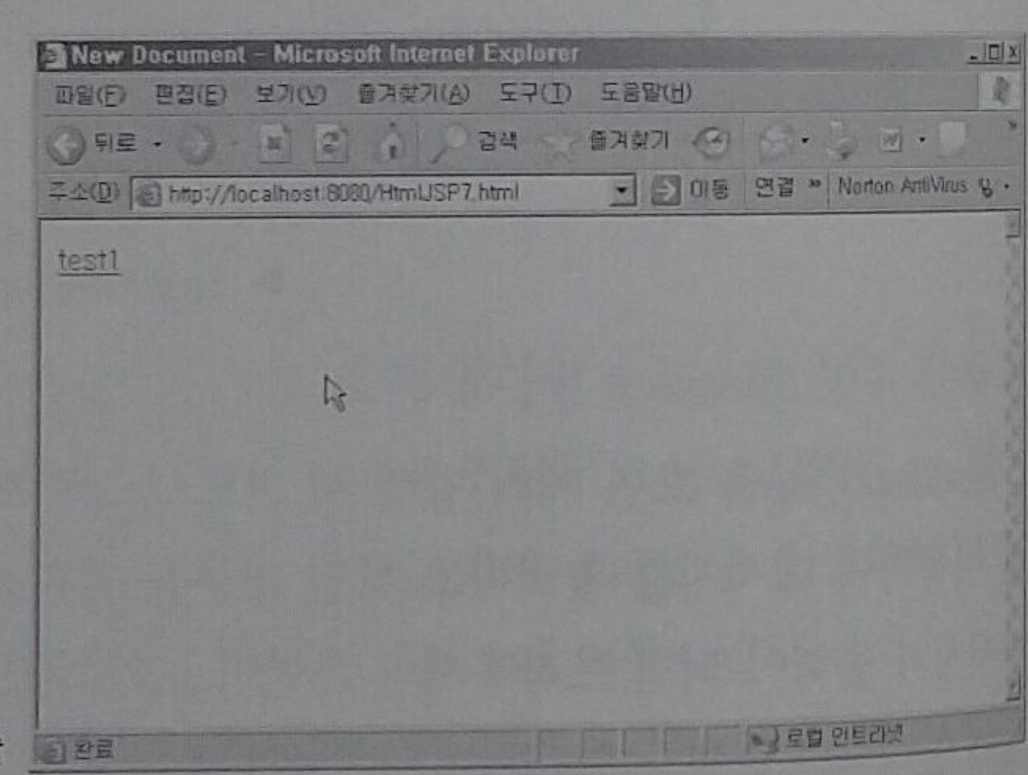
▶ onClick() 예제

링크된 test라는 글자를 클릭하면 go_test1()이라는 function을 호출하게 한다.



▶ 스크립트의 작성

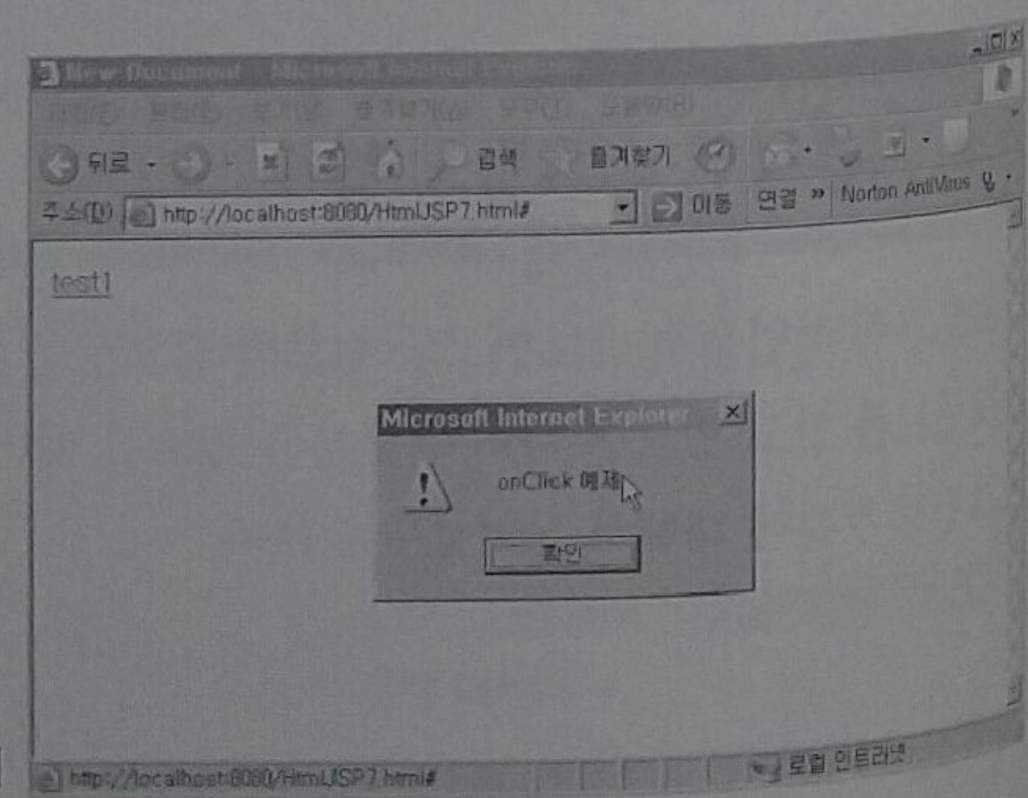
브라우저를 열고 HtmlJSP7.html을 호출한다.



▶ HtmlJSP7.html 호출

test 글자를 클릭하면 메시지를 띄우게 호출한다.

주로 onClick() 이벤트는 입력 페이지에서 입력 값을 체크하고 그 체크된 입력 값을 결과처리 페이지로 파라메타의 값을 전달할 때 많이 사용된다.



▶ 메시지 처리

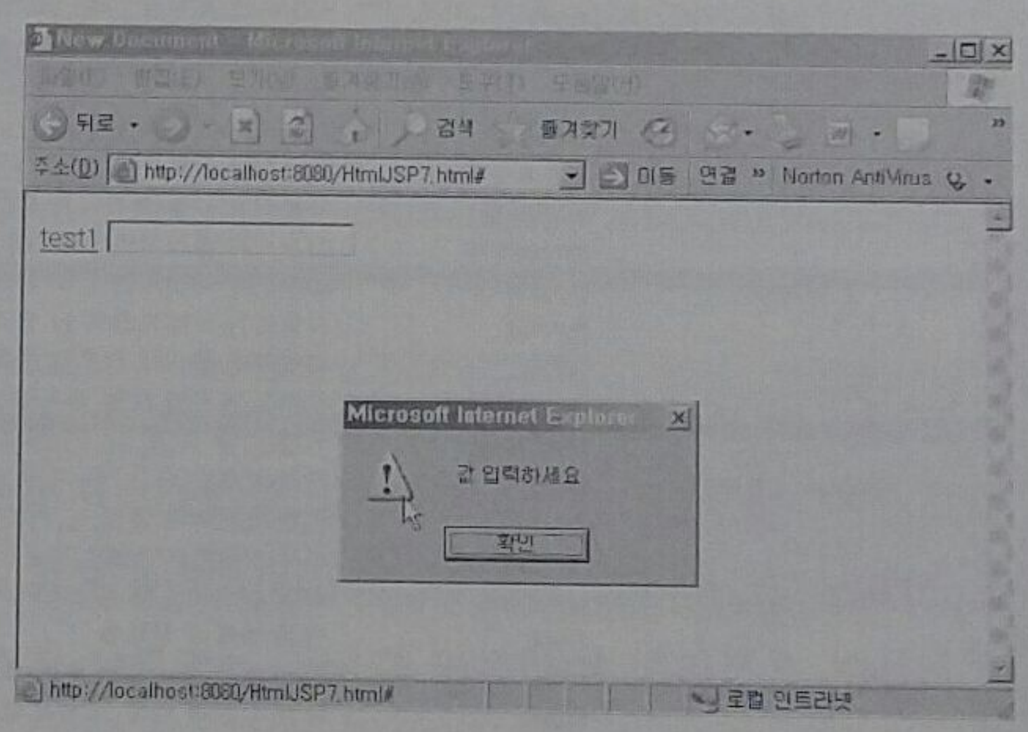
focus()

입력 페이지에 특정한 입력 오브젝트 또는 리턴되는 오브젝트에 포커스를 주는 이벤트이다. 주로 입력 값을 체크하여 미 입력 값을 주었을 경우 메시지를 띄우고 사용자 편의성을 위해 해당 입력 오브젝트에 포커스를 준다. 대신 form에 대한 이름을 주어야 해당 오브젝트를 쉽게 인지할 수 있다.

```
<HTML>
<HEAD>
<TITLE> New Document </TITLE>
<script language="javascript">
  function go_test1()
  {
      alert("onClick 예제");

      if(document.frm.test.value=="")
      {
         alert("값 입력하세요");
         document.frm.test.focus();
      }
  }

</script>
</HEAD>
<BODY>
<form name="frm" method="post">
<a href="#" onClick="javascript:go_test1()">test1</a>
<input type="text" name="test" >
</form>
```

▶ HtmlJSP7.html에 소스 추가

저장하고 브라우저에서 새로 고침 또는 HtmlJSP7.html를 호출하여 test를 클릭하면 첫 번째 메시지 이후 입력 여부에 대한 메시지를 출력한다.



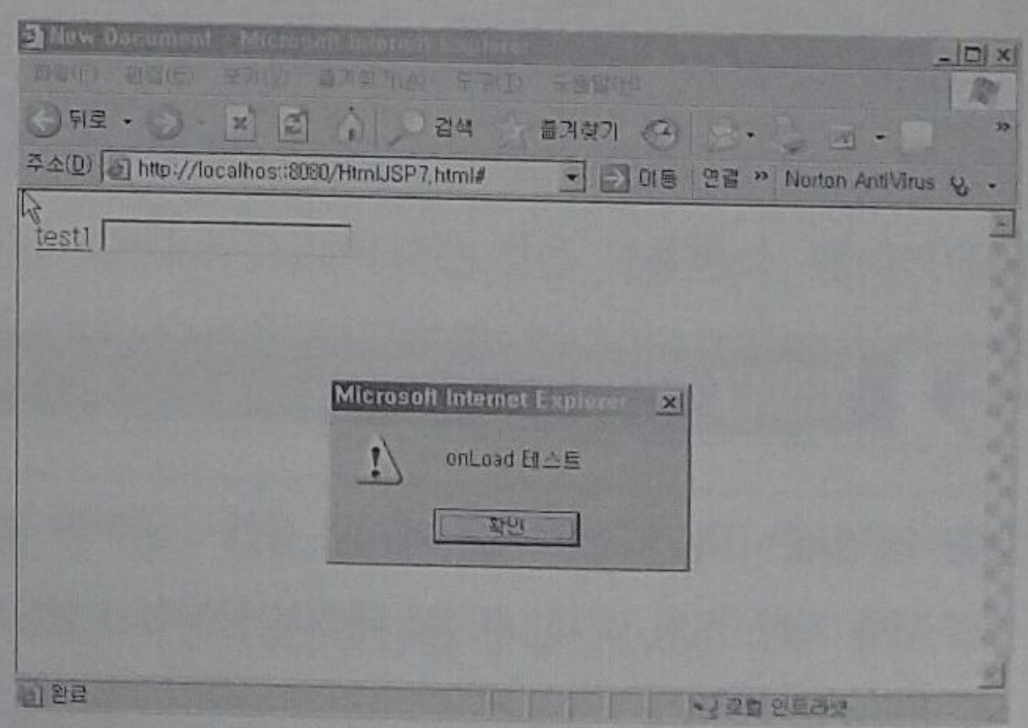
▶ 결과

onLoad()

페이지가 브라우저에 처음 띄워질 때 또는 새로 고침을 하였을 경우 폼을 읽어 들이기 전에 가장 먼저 실행되는 스크립트라고 볼 수 있다.

```
  }

  function go_test2()
  {
      alert("onLoad 테스트");
  }

</script>
</HEAD>
<BODY onload="javascript:go_test2()">
<form name="frm" method="post">
<a href="#" onClick="javascript:go_test1()">test1</a>
<input type="text" name="test" >
</form>
</BODY>
</HTML>
```

▶ onLoad() 이벤트 추가



▶ 실행결과

onLoad() 이벤트는 팝업창 작업이나 또는 앞 예제처럼 메시지 처리를 할 때 주로 사용된다. 이외 기타 이벤트 함수 등이 있다.

TIP

스크립트 객체와 이벤트

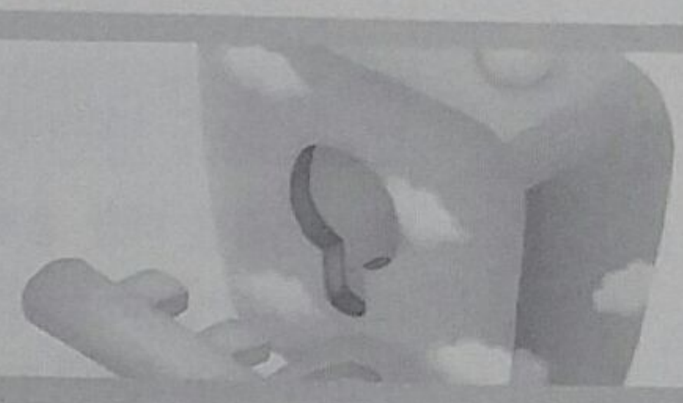
스크립트의 객체와 여러 가지 이벤트를 검색하여 따로 모아둔 것으로 참고하기 바란다.

HTML 태그	JavaScript 이벤트	설명
다양	mouseMove	마우스 이동
	Click	마우스로 링크를 클릭
	dbClick	마우스를 링크위에서 더블클릭
	mouseDown	마우스 버튼을 누름
	mouseUp	마우스 버튼을 놓음
<A>..</A>	mouseOver	마우스를 링크위로 이동
	mouseOut	링크 위에 있던 마우스를 링크 밖으로 이동
	keyDown	사용자가 키를 누름
	keyUp	사용자가 키를 놓음
	keyPress	사용자가 키를 눌렀다가 놓음
	abort	사용자 액션으로 인해 이미지 로딩 작업을 중단함
	error	이미지 로딩하는 동안 에러 발생
<IMG>	load	이미지가 로드되고 화면에 나타남
	keyDown	사용자가 키를 누름
	keyUp	사용자가 키를 놓음
	keyPress	사용자가 키를 눌렀다가 놓음
	mouseOver	마우스가 클라이언트측 이미지맵의 한 영역으로 이동함
<AREA>	mouseOut	마우스가 이미지맵 영역 밖으로 이동
	dbClick	사용자가 이미지맵의 한 영역을 더블클릭함
	Click	사용자가 문서의 본문을 클릭
	dbClick	문서의 본문을 더블 클릭함
	keyDown	키를 누름
<BODY>..</BODY>	keyUp	키를 놓음
	keyPress	키를 눌렀다가 놓음
	mouseDown	마우스 버튼을 누름
	mouseUp	마우스 버튼을 놓음
	Click	제출 버튼이 클릭됨
<INPUT TYPE="submit">	blur	제출 버튼에서 입력 포커스가 사라짐
	focus	제출 버튼에 입력 포커스 생김
	Click	리셋 버튼이 클릭됨
<INPUT TYPE="reset">	blur	리셋 버튼에서 포커스가 사라짐
	focus	리셋 버튼에서 포터스가 놓임
	Click	라디오 버튼이 클릭
<INPUT TYPE="radio">	blur	라디오 버튼에서 입력 포커스가 사라짐
	focus	라디오 버튼에 입력 포커스 생김
	Click	체크 박스가 클릭
<INPUT TYPE="checkbox">	blur	체크 박스에서 입력 포커스가 사라짐
	focus	체크 박스에 입력 포커스 놓임
	blur	파일 업로드 폼 요소에서 입력 포커스 사라짐
<INPUT TYPE="file">	change	사용자가 업로드될 파일을 선택
	focus	파일 업로드 폼 요소에 입력 포커스 놓임
	blur	선택 요소가 현재 입력 포커스 잃음
<SELECT>..</SELECT>	change	선택 요소가 변경되어 입력 포커스가 사라짐
	focus	선택 요소에 현재 입력 포커스가 놓임

맺음말

이번 장에서는 TMJ의 맛보기를 예제를 통해서 공부해 보았다. 이제 다음 장부터는 본격적으로 Mysql과 연동하여 데이터를 저장, 조회, 삭제, 수정에 대해서 공부하고 앞으로 Tip과 연동하여 여러 가지 예제를 통해 기술을 습득할 수 있는 기초가 된다.

09 MySql과 연동하기



안녕하세요

8장에서 TMJ에 대한 기본적인 공부였다면 이번 장에서는 본격적인 TMJ의 시작이라고 할 수 있다. 이번 장에서는 설치된 MySql과의 연동을 통해서 데이터를 조작하는 방법에 대해서 공부하고 예제를 통해서 그 과정을 습득하게 된다.

MySql과의 연동

설치된 데이터베이스 MySql과 연동하기 위해서는 먼저 JDBC를 설치해야 한다. JDBC는 자바 프로그램 내에 데이터베이스 질의문 즉 쿼리를 실행하기 위한 API(Application programming interface)이다. 따라서 JDBC를 사용하면 어떠한 데이터베이스건 상관없이 자바와 연동하여 쿼리문을 작성하고 이를 사용할 수 있다.

데이터베이스 연결 방법은 3가지가 있는데 첫 번째는 바로 데이터베이스에 연결하여 프로그램을 작성하는 방법과 둘째는 톰캣의 server.xml 내에 DBCP(DB Connection Pool)로 선언하여 사용하고 마지막으로는 DBConnectionManager를 사용하는 방법이다.

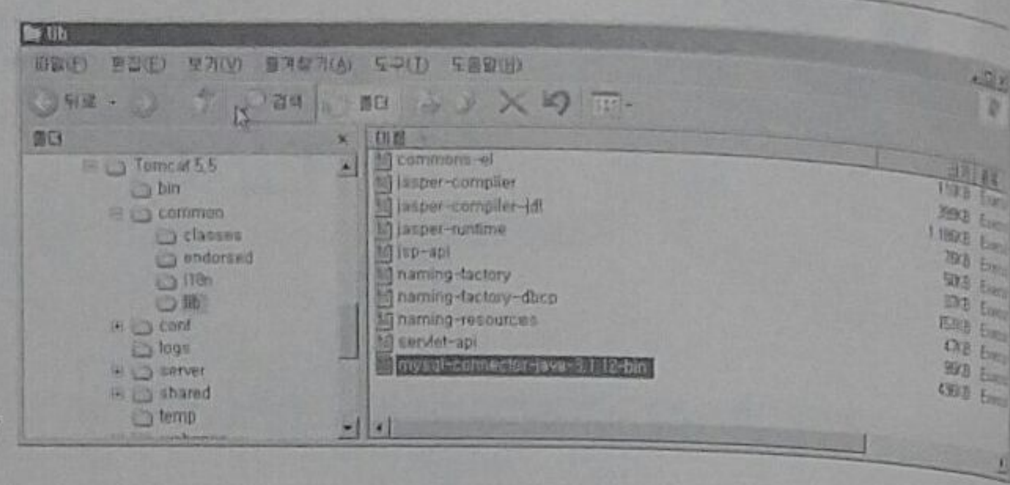
예전에는 ConnectionManager를 많이 사용했으나 현재는 톰캣 자체의 성능의 향상으로 DBCP를 사용한다. DBCP는 보통의 경우 데이터베이스에 연결을 하고 끊어버리게 되는데 데이터베이스에 연결하는 과정은 시간이 많이 소요 된다.

이러한 문제점을 해결하기 위해서 데이터베이스 Connection Pool이라는 것을 사용한다. 데이터베이스 Connection Pool은 Pool 매니저가 일정의 Connection을 연결하고 있다가 요청이 들어오면 Connection을 할당해 주고 없다면 기다린다.

즉 요청한 Connection을 다 쓰면 다시 반납하고 반납 받은 Connection을 재분배한다. 따라서 속도면이나 퍼포먼스 부분을 향상 시킨다. 이번 장에서는 기본적인 연결방법을 통한 MySql과의 연동 방법에 대해서 공부한다.

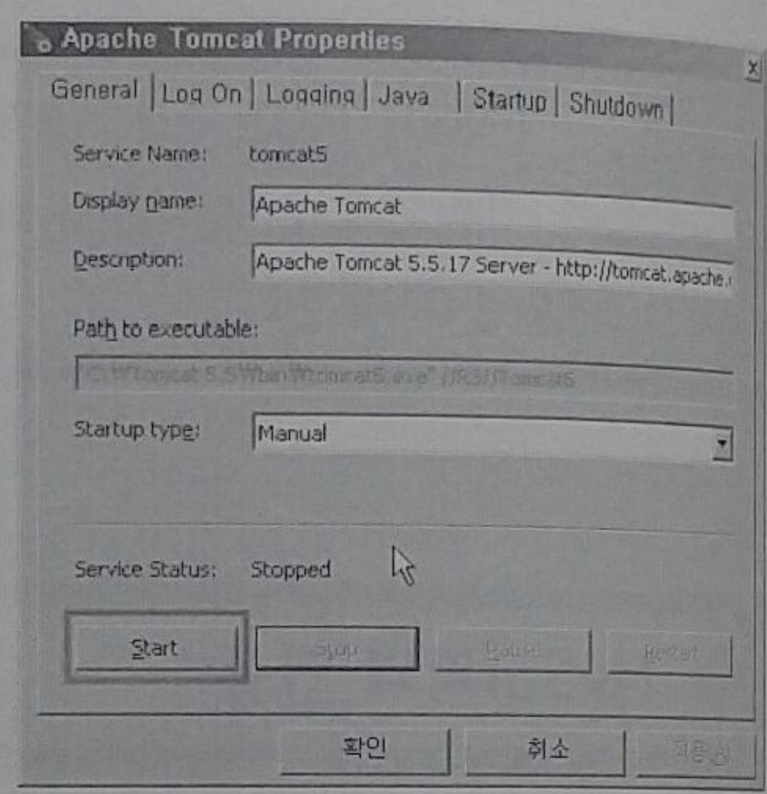
따라하기 mysql과의 연동 예제

❶ 도서출판 대림 홈페이지 자료실에 있는 mysql-connector-java-3.1.12-bin.jar 파일을 다운받은 후 설치된 Tomcat 폴더의 common의 lib에 복사한다.



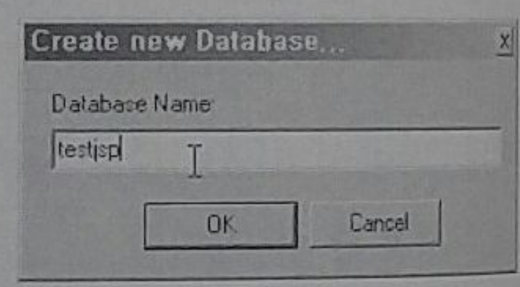
▶ mysql JDBC의 복사

❷ Tomcat을 가동시킨다.



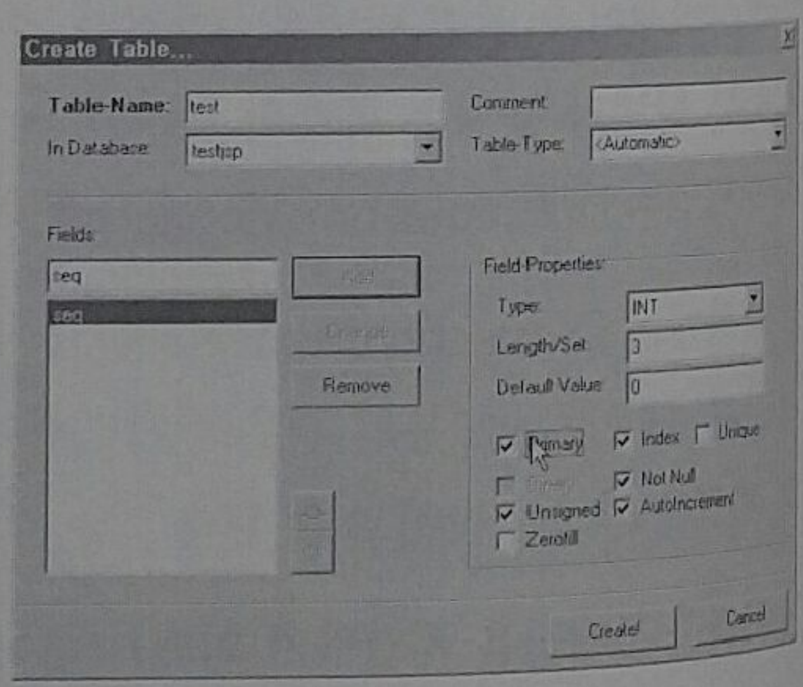
▶ Tomcat의 start

❸ mysql_front에서 테스트 할 데이터베이스를 생성한다.

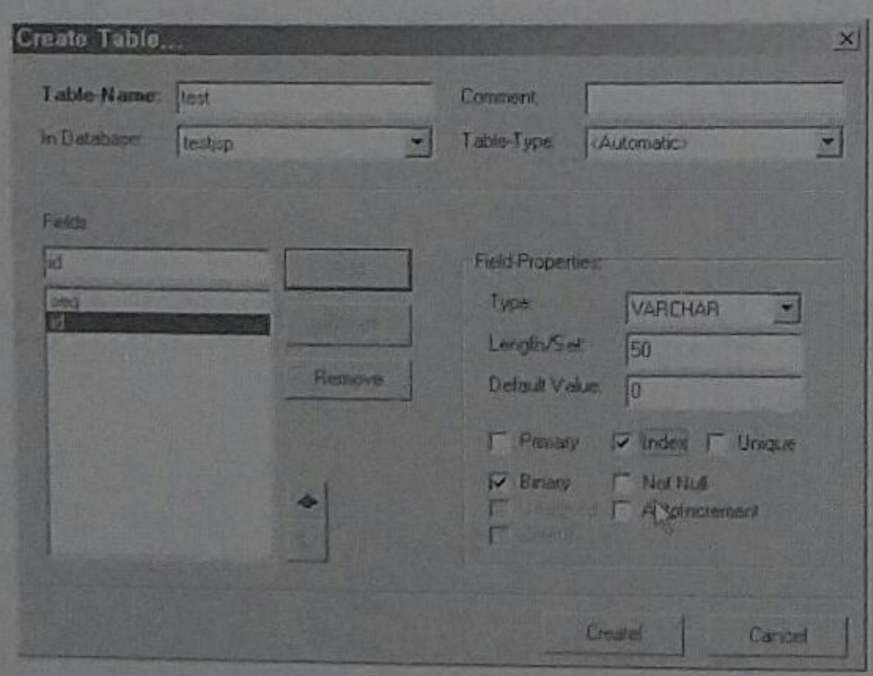


▶ testjsp DB생성

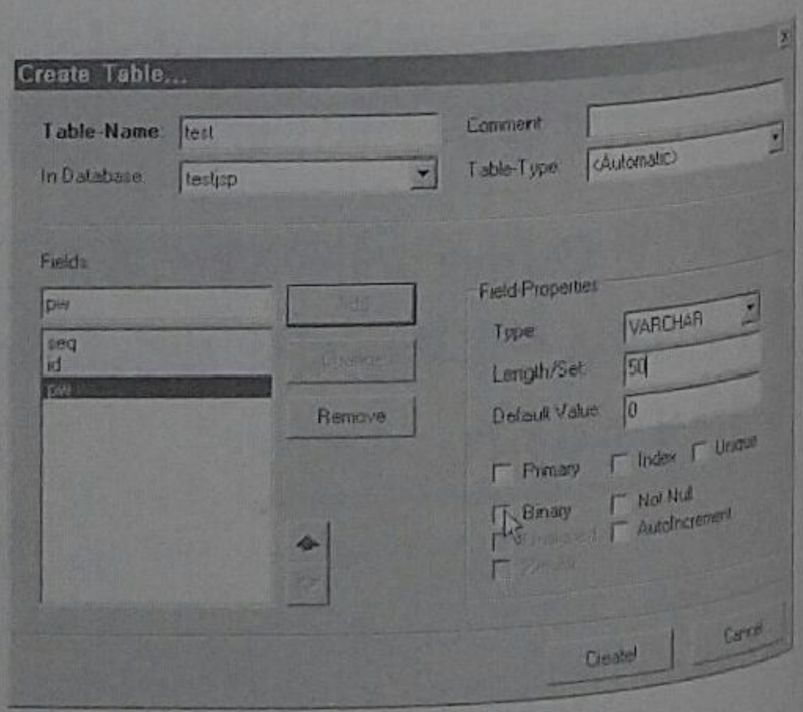
❹ testjsp 내 test001이라는 테이블을 생성하고 시퀀스, id, pw의 칼럼을 생성한다.



▶ 시퀀스 칼럼 생성



▶ id 칼럼 생성



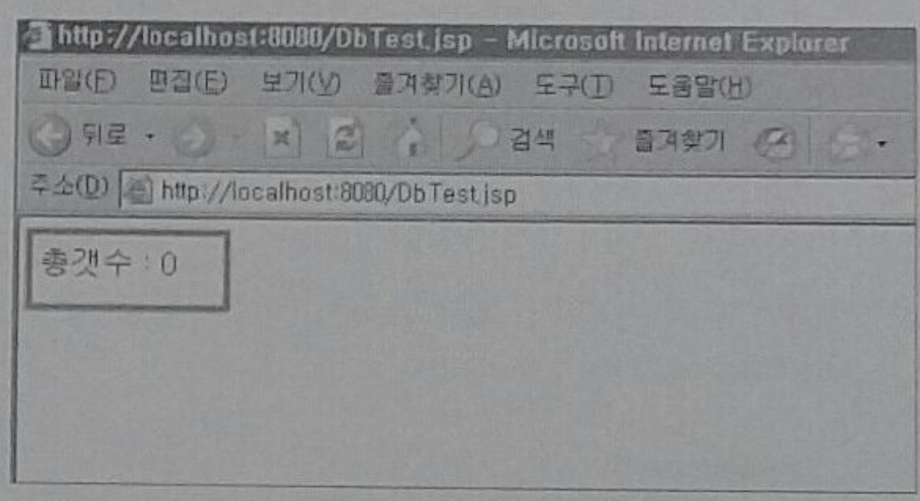
▶ pw 칼럼 생성

❺ 데이터베이스 접속 테스트용 JSP를 작성한다.

```
<%@ page language="java" import="java.sql.*, java.lang.*, java.util.* " contentType=
"text/html;charset = euc-kr"%>
<%
int count=0;

    try
    {
        Class.forName("org.gjt.mm.mysql.Driver");
        Connection Conn =
        DriverManager.getConnection("jdbc:mysql://localhost/test","root","");

        Statement stmt = Conn.createStatement();
        ResultSet rs = stmt.executeQuery("select count(*) from test");
        if(rs.next())
        {
            count=rs.getInt(1);
        }
        rs.close();
        stmt.close();
    }
    catch(SQLException e)
    {
    }
%>
총갯수 : <%=count%>
```

▶ DbTest.jsp 작성

❻ 브라우저에서 DbTest.jsp 파일을 호출한다. 0이 나오면 테이블과 연결에 성공한 것이다.



▶ DbTest.jsp의 실행

데이터베이스에 연결이 되었다면 생성된 test 테이블에 입력 페이지를 만들어 insert하는 JSP를 작성한다.

기본예제-데이터 저장하기

따라하기 데이터 저장

❶ 아이디/ 패스워드를 입력하는 입력 페이지를 출력하는 Dbinsert_in.jsp을 작성한다.

```
<%@ page language="java" contentType="text/html; charset=euc-kr" %>
<%
  //결과 메시지를 처리하기 위한 변수 값 설정 및 파라메타 받기
  String msg=request.getParameter("msg");
%>

  <html>
  <head>
  <title> new document </title>
  <script language="javascript">
  //저장 시 아이디와 암호 입력 여부를 체크하는 스크립트
```

다음 페이지에 계속

```
function go_save()
{
   var frm = document.formm;
   if(frm.id.value =="")
    {
       alert("아이디를 입력 하세요");
      frm.id.focus();
    }
    else if(frm.pw.value=="")
    {
      alert("암호를 입력하세요");
     frm.pw.focus();
    }
    else
    {
      if(confirm("저장하시겠습니까?"))
       {
          frm.action="./Dbinsert_save.jsp";
          frm.submit();
       }
    }
}

//페이지 재 호출시 파라메타를 받아 메시지 처리하는 스크립트
function show_msg(msg)
{
  if(msg ==1)
   {
     alert("저장이 되었습니다.");
   }
   else if( msg ==2)
   {
     alert("오류가 발생되었습니다.");
    }
}
</script>
</head>
<body onload="show_msg( '<%=msg%>' );">
<form name="formm" method="post" >
  아 이 디:<input type="text" name="id"><br>
  패스워드:<input type="password" name="pw"><br>
  <input type="button" value="확인" onclick="javascript:go_save();">
</form>
</body>
</html>
```

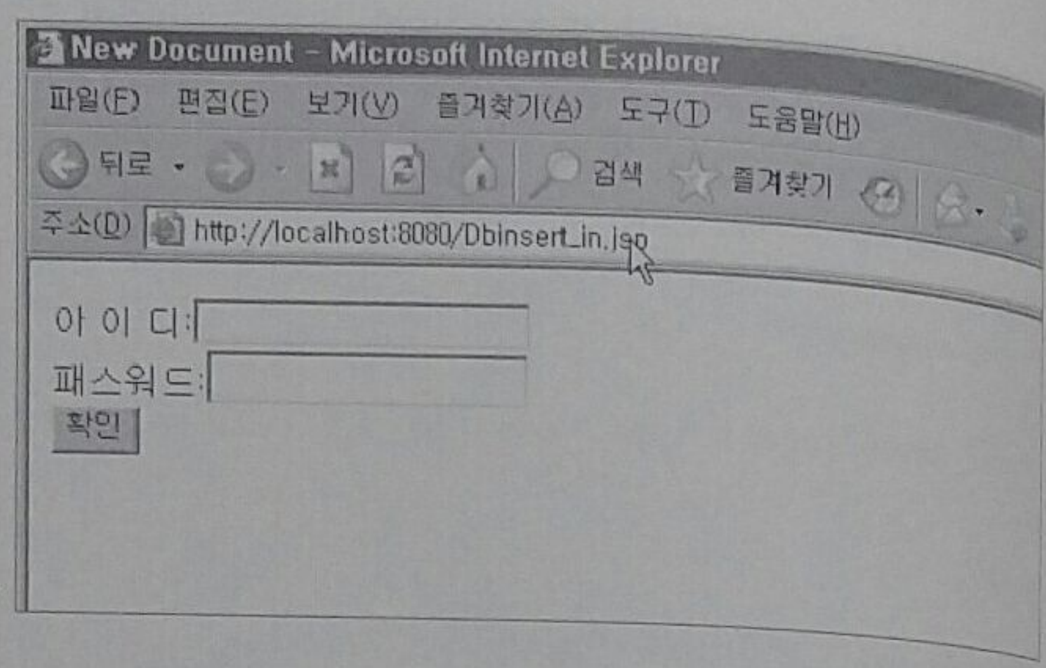
❷ Dbinsert_in.jsp에서 파라메타를 받아 test 테이블에 저장하는 Dbinsert_save.jsp을 작성한다.

```
<%@ page language="java" import="java.sql.*, java.lang.*, java.util.* "
                    contentType= "text/html;charset = euc-kr"%>
<%

//아이디를 받아들이는 변수
String id=request.getParameter("id");
//암호를 받아들이는 변수
```

다음 페이지에 계속

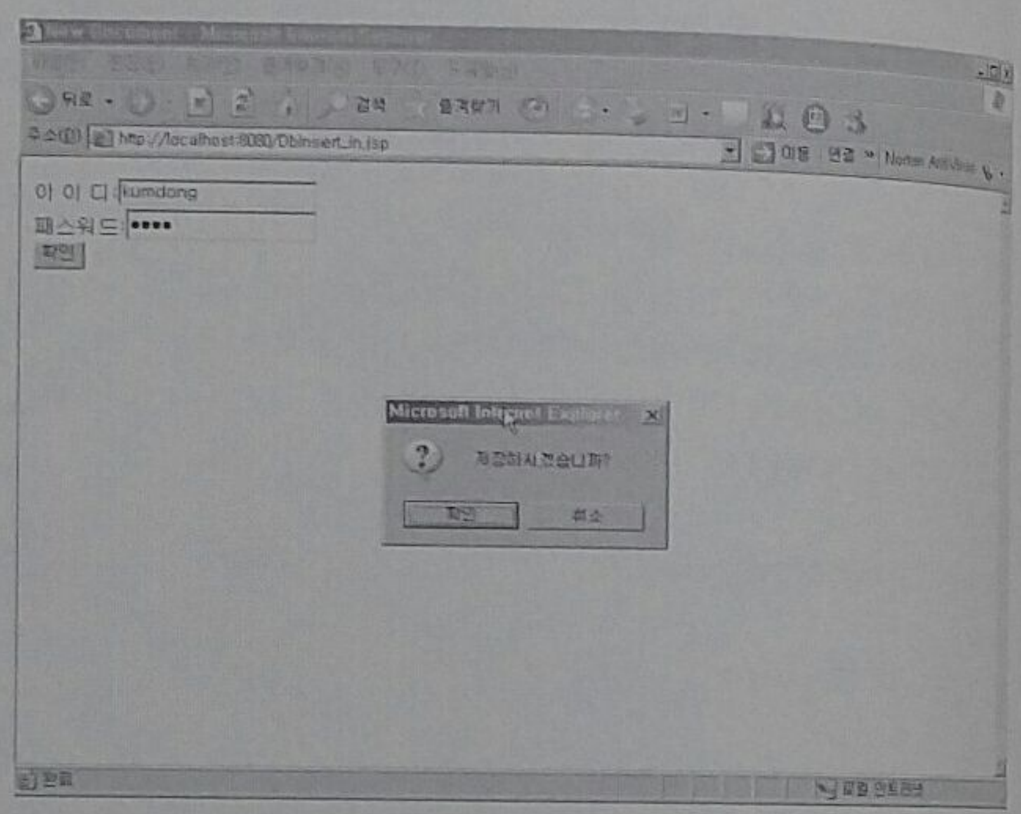
```
String pw=request.getParameter("pw");
//저장 쿼리를 작성할 변수
String sql_save="";
//결과를 리턴하려는 변수
int message=0;

sql_save="insert into test (id,pw) values ('"+id+"','"+pw+"')";

try
{
  //mysql 드라이버의 연결
  Class.forName("org.gjt.mm.mysql.Driver");
  Connection Conn =
  DriverManager.getConnection("jdbc:mysql://localhost/testjsp","root","");
  Statement stmt = Conn.createStatement();

 //executeUpdate를 사용하면
 //리턴 값이 정상적일 때는 1 비정상적일 때는 -1을 리턴
 message=stmt.executeUpdate(sql_save);
 stmt.close();
 Conn.close();
}
catch(SQLException e)
{
 out.println(e);
}
catch(Exception e1)
{
 out.println(e1);
}

//insert문을 실행하고 반환되는 리턴 값에 따라 if을 실행한다.
if(message > 0)
{
%>
<form name="frm" method="post" action="./Dbinsert_in.jsp">
<input type="hidden" name="msg" value="1">
</form>
<script language="javascript">
 document.frm.submit();
</script>
<%
}
else
{
%>
<form name="frm" method="post" action="./Dbinsert_in.jsp">
<input type="hidden" name="msg" value="2">
</form>
<script language="javascript">
 document.frm.submit();
</script>
<%
}
%>
```

❸ 브라우저에서 Dbinset_in.jsp를 호출한다.



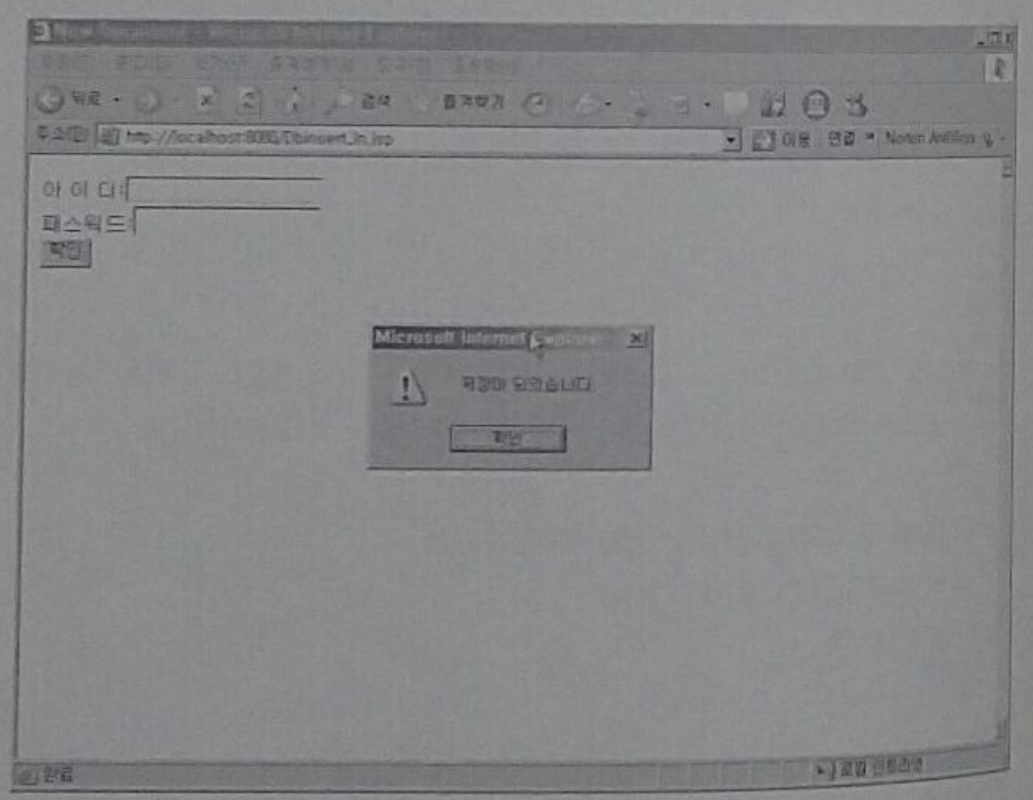
▶ Dbinsert_in.jsp 호출

❹ 값을 입력하고 확인 버튼을 클릭한다.



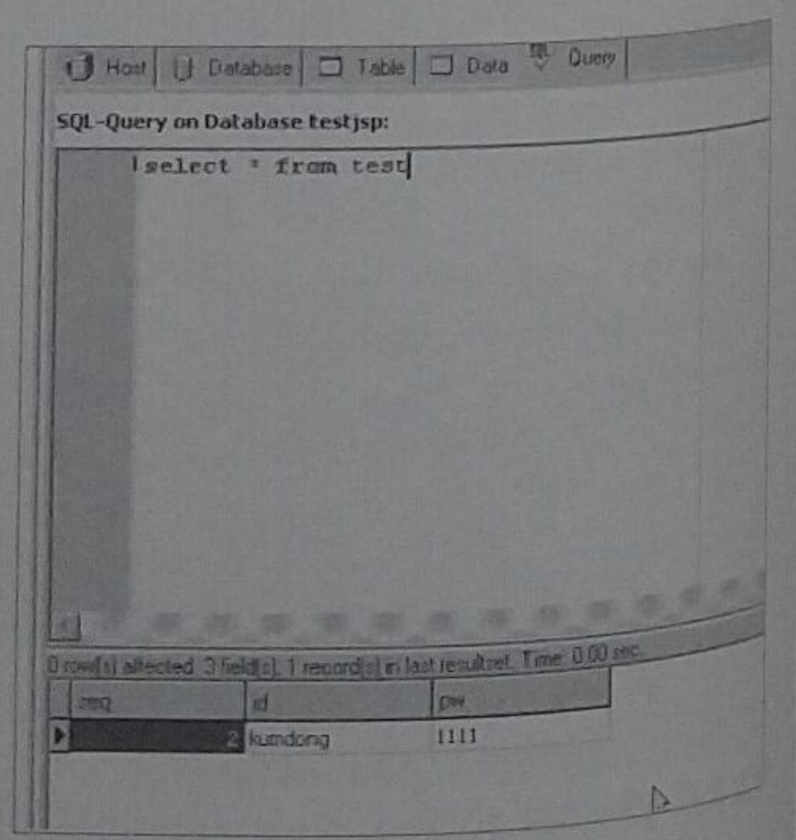
▶ 값 입력

❺ 성공적으로 저장된 메시지를 출력한다.



▶ 저장성공

❻ MySql-Front에서 저장된 데이터를 확인한다.



▶ 데이터베이스 확인

MySql에서 저장이 이루어졌다면 입력화면을 출력하는 것이 아니라 이제 저장된 데이터를 볼 수 있도록 리스트를 출력하도록 한다.

기본예제- 데이터 리스트

따라하기 데이터 리스트

❶ 데이터베이스에 저장된 데이터를 읽어 출력하는 Dbinsert_list.jsp 파일을 작성한다.

```
<%@ page language="java" import="java.sql.*, java.lang.*, java.util.* "
                    contentType= "text/html;charset = euc-kr"%>
<%

//Connection, 쿼리, 결과값을 가지고 오는 선언
Connection Conn=null;
Statement stmt=null;
ResultSet rs =null;

//리스트를 가지고 오는 쿼리 변수
String db_list="";
db_list="select * from test";

try
{
 //mysql 드라이버의 연결
 Class.forName("org.gjt.mm.mysql.Driver");
 Conn= DriverManager.getConnection("jdbc:mysql://localhost/testjsp","root","");
 stmt = Conn.createStatement();
%>
<TABLE border="1">
 <TR>
   <TD>번호</TD>
   <TD>아이디</TD>
   <TD>패스워드</TD>
</TR>
<%
//쿼리 실행하고 결과 값을 while문으로 출력
rs = stmt.executeQuery(db_list);
while(rs.next())
{
%>
  <tr>
   <td><%=rs.getString(1)%></td>
   <td><%=rs.getString(2)%></td>
   <td><%=rs.getString(3)%></td>
  </tr>
<%
}
rs.close();
stmt.close();
Conn.close();
}
catch(SQLException e)
```

다음 페이지에 계속

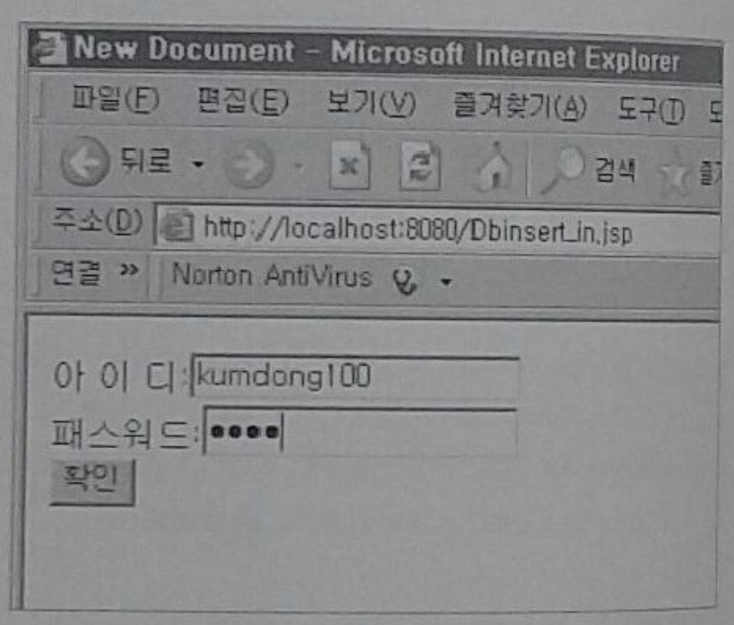
```
{
 out.println(e);
}
catch(Exception e1)
{
 out.println(e1);
}
%>
</TABLE>
```

❷ Dbinsert_save.jsp 파일을 열고 저장이 성공적으로 이루어지면 이동하는 action을 Dbinsert_list.jsp 파일로 바꿔준다.

```
if(message > 0)
{
%>
<form name="frm" method="post" action="./Dbinsert_list.jsp">
<input type="hidden" name="msg" value="1">
</form>
<script language="javascript">
    document.frm.submit();
</script>
<%
}
else
{
```

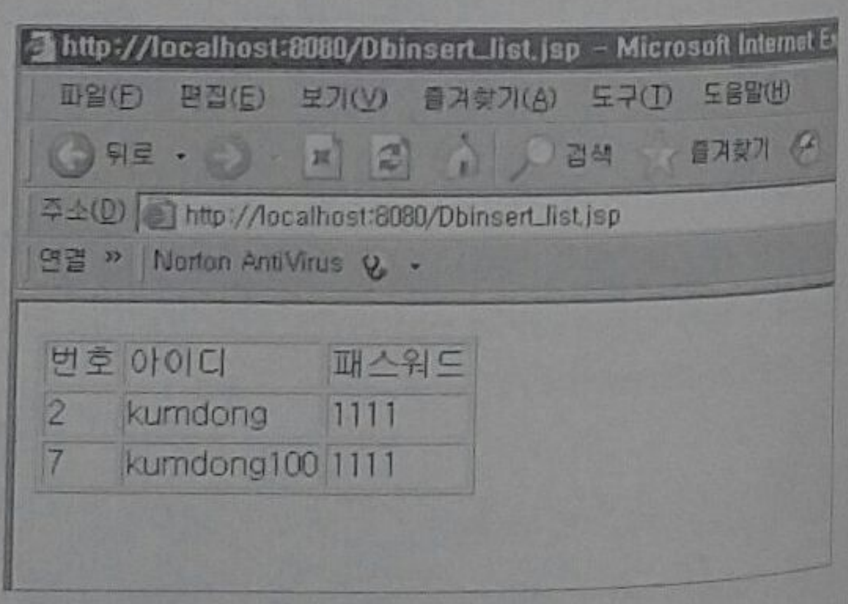
▶ 리턴페이지의 변동

❸ 브라우저를 열고 Dbinsert_in.jsp를 실행하고 값을 입력한다.



▶ 아이디/ 패스워드 입력

❹ 확인 버튼을 클릭하면 저장과 동시에 입력된 리스트로 이동된다.



▶ 저장 후 리스트로의 이동

입력 값을 저장하고 입력된 저장 데이터를 리스트로 출력해 보았다. 그런데 뭔가가 아쉬움이 남는 것을 느끼게 된다. 데이터를 저장하려면 다시 Dbinsert.jsp 파일을 호출해야 한다. 따라서 입력 폼과 리스트 폼을 같은 화면에 보여주고자 한다면 저장되는 Dbinsert_in.jsp 파일과 리스트를 출력해 주는 Dbinsert_list.jsp 파일을 합쳐야만 할 것이다.

기본예제- 데이터 저장과 리스트를 한 화면에서의 처리

따라하기 데이터 저장과 리스트를 한 화면에서의 처리하기

❶ 이미 작성되어진 Dbinsert_in.jsp 파일 내에 리스트를 출력하는 부분을 삽입한다.

```
<%@ page language="java" import="java.sql.*, java.lang.*, java.util.* "
                        contentType= "text/html;charset = euc-kr"%>
<%

//Connection, 쿼리, 결과값을 가지고 오는 선언
Connection Conn=null;
Statement stmt=null;
ResultSet rs =null;

//결과 메시지를 처리하기 위한 변수 값 설정 및 파라메타 받기
String msg=request.getParameter("msg");

//리스트를 가지고 오는 쿼리 변수
String db_list="select * from test";

//리스트에 연번 증가를 위한 변수
int num=1;
%>
<HTML>
<HEAD>
<script language="javascript">

//저장 시 아이디와 암호 입력 여부를 체크하는 스크립트
function go_save()
{
  var frm = document.formm;
  if(frm.id.value =="")
  {
    alert("아이디를 입력하세요");
    frm.id.focus();
  }
  else if(frm.pw.value=="")
  {
    alert("암호를 입력하세요");
    frm.pw.focus();
   }
  else
  {
    if(confirm("저장하시겠습니까?"))
    {
      frm.action="./Dbinsert_save.jsp";
      frm.submit();
    }
  }
}

//페이지 재 호출시 파라메타를 받아 메시지를 처리하는 스크립트
function show_msg(msg)
```

다음 페이지에 계속 ➲

```
{
  if(msg =1)
  {
    alert("저장이 되었습니다.");
  }
  else if( msg =2)
  {
    alert("오류가 발생되었습니다.");
  }
}
</script>
</head>
<body onload="show_msg( '<%=msg%>' );">
<form name="formm" method="post" >
아 이 디:<input type="text" name="id"><br>
패스워드:<input type="password" name="pw"><br>
<input type="button" value="확인" onclick="javascript:go_save();">
<br>
<%
try
{
 //mysql 드라이버의 연결
 Class.forName("org.gjt.mm.mysql.Driver");
 Conn= DriverManager.getConnection("jdbc:mysql://localhost/testjsp","root","");
 stmt = Conn.createStatement();
%>
<table border="1">
<tr>
 <td>번호</td>
 <td>아이디</td>
 <td>패스워드</td>
</tr>
<%
//쿼리 실행하고 결과 값을 while문으로 출력
rs = stmt.executeQuery(db_list);
while(rs.next())
{
%>
 <tr>
  <td><%=num%></td>
  <td><%=rs.getString(2)%></td>
  <td><%=rs.getString(3)%></td>
 </tr>
<%
num++;
}
rs.close();
stmt.close();
Conn.close();

}
catch(SQLException e)
{
 out.println(e);
}
catch(Exception e1)
{
 out.println(e1);
```

다음 페이지에 계속

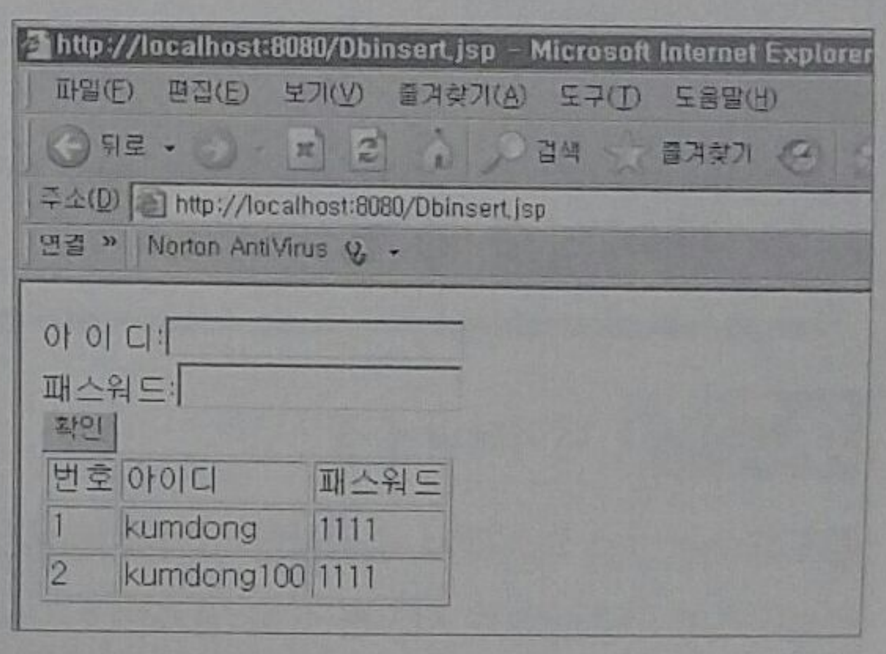
```
}
//저장된 데이터가 하나도 없을 시에 출력되는 메시지
if(num =1)
{
%>
<tr>
<td  colspan="3">저장된 데이터가 없습니다.</td>
</tr>
<%
}
%>
</table>
</form>
</body>
</html>
```

❷ Dbinsert.jsp로 저장 후 Dbinsert_save.jsp 파일에서 저장 시 또는 에러발생 시 반환되는 페이지의 action을 변경한다.

```
if(message > 0)
{
%>
<form name="frm" method="post" action="./Dbinsert.jsp">
<input type="hidden" name="msg" value="">
</form>
<script language="javascript">
    document.frm.submit();
</script>
<%
}
else
{
%>
<form name="frm" method="post" action="./Dbinsert.jsp">
<input type="hidden" name="msg" value="">
</form>
<script language="javascript">
    document.frm.submit();
</script>
<%
}
%>
```

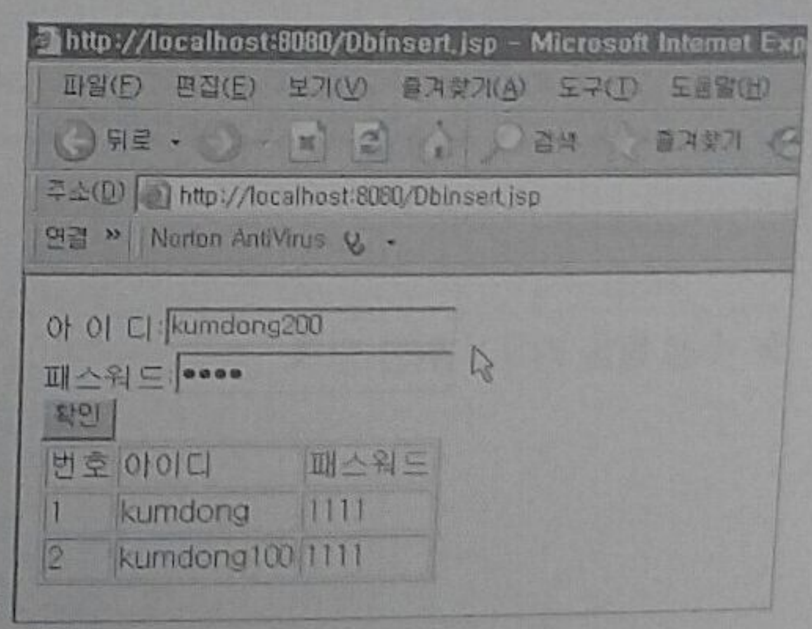
▶ 페이지 이동 변경

❸ 브라우저를 열고 Dbinsert.jsp를 호출한다.



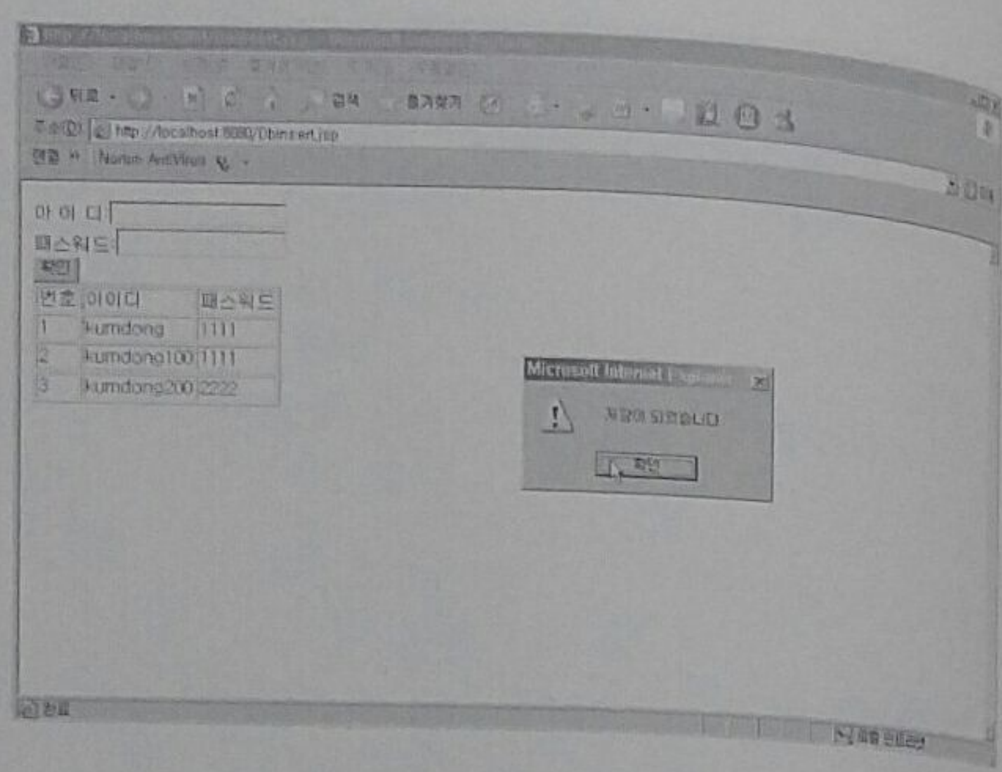
▶ Dbinsert.jsp 의 호출

❹ 입력 값을 입력하고 확인 버튼을 클릭한다.



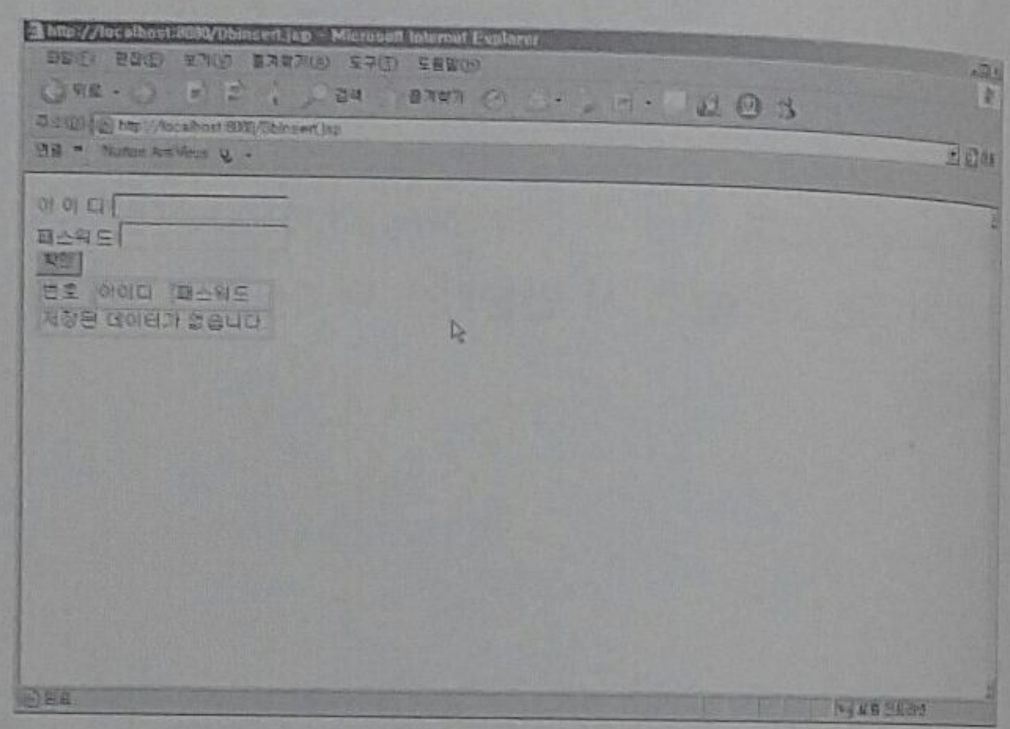
▶ 입력 값

❺ 입력이 완료가 되면 입력된 리스트가 입력화면과 같이 출력이 된다.



▶ 입력 결과

❻ 데이터베이스에서 데이터를 삭제하고 Dbinsert.jsp를 실행하면 num==1에 의해 메시지를 출력한다.



▶ 데이터 존재 유무에 대한 메시지

데이터의 저장과 리스트 확인 후 잘못 입력된 데이터의 삭제와 수정은 필수이므로 예제를 통해서 수정과 삭제를 공부하도록 하자

데이터의 수정

따라하기 데이터의 수정

❶ Dbinsert.jsp 파일에서 수정화면 팝업을 띄우기 위한 링크를 작성한다. 스크립트에서 값을 넘겨주는데 이 값은 데이터베이스에 PK(키값)로 지정된 값을 넘겨준다.

```
%>
        <tr>
            <td>
                <td><%=num%></td>
                <td><a href="#" onclick="javascript:go_mod('<%=rs.getString(1)%>')"><%=rs.getString(2)%></a></td>
                <td><%=rs.getString(3)%></td>
            </td>
<%
            num++;
            }
            rs.close();
```

▶ 수정 창을 띄우기 위한 링크

❷ 수정 창을 열기 위한 스크립트를 작성한다.

```
function go_mod(seq)
{
    var url="./Dbinsert_mod.jsp?seq="+seq;
  window.open(url, '_blank', 'menubar=no,resizable=no,toolbar=no,scrollbars=yes,status=no,width=420,height=310');
}

</script>
</HEAD>
```

▶ 스크립트의 작성

❸ 데이터의 수정을 위해 Dbinsert_mod.jsp 파일을 작성한다.

```
<%@ page language="java" import="java.sql.*, java.lang.*, java.util.* "
                                contentType= "text/html;charset = euc-kr"%>
<%
//스크립트에서 넘겨오는 파라메타를 받는 변수
String seq=request.getParameter("seq");
%>
<html>
<head>
<script language="javascript">

//수정 시 아이디와 암호 입력 여부를 체크하는 스크립트
function go_mod(seq)
{
  var frm = document.formm;
  if(frm.id.value =="")
  {
   alert("아이디를 입력하세요");
   frm.id.focus();
  }
  else if(frm.pw.value=="")
  {
   alert("암호를 입력하세요");
   frm.pw.focus();
  }
  else
  {
    if(confirm("수정하시겠습니까?"))
    {
      frm.action="./Dbinsert_save.jsp?seq="+seq+"&save_mod_del=2";
      frm.submit();
    }
  }
}
```

다음 페이지에 계속 ◎

```
</script>
</head>
<body>
<form name="formm" method="post" >
아 이 디:<input type="text" name="id"><br>
패스워드:<input type="password" name="pw"><br>
<input type="button" value="확인" onclick="javascript:go_mod(<%=seq%>);">
</form>
</body>
</html>
```

여기서 눈여겨 봐야할 부분은 다음이다.

```
frm.action="./Dbinsert_save.jsp?seq="+seq+"&save_mod_del=2";
```

seq는 수정을 위한 PK(키값)의 값이지만 그 뒤에 save_mod_del의 값은 저장할 것인지 수정할 것인지 삭제할 것인지를 구별하는 구분자가 된다.

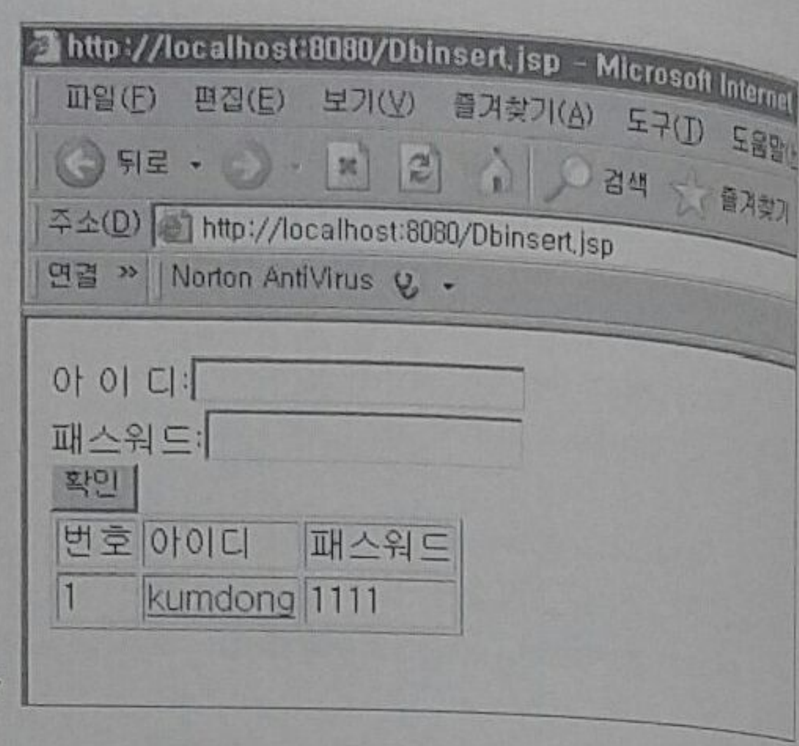
❹ Dbinsert_save.jsp 파일을 열고 다음과 같이 수정을 한다.

```
<%@ page language="java" import="java.sql.*, java.lang.*, java.util.* "
                                contentType= "text/html;charset = euc-kr"%>
<%
// 수정 또는 삭제를 위한 Pk를 받는 변수
String seq=request.getParameter("seq");

//저장, 수정, 삭제 구분
String save_mod_del=request.getParameter("save_mod_del");

//아이디를 받아들이는 변수
String id=request.getParameter("id");

//암호를 받아들이는 변수
String pw=request.getParameter("pw");

//저장 쿼리를 작성할 변수
String sql_save="";

//수정 쿼리를 작성할 변수
String sql_update="";

//삭제 쿼리를 작성할 변수
String sql_delete="";

//결과를 리턴하려는 변수
int message=0;

sql_save="insert into test (id,pw) values ('"+id+"','"+pw+"')";
sql_update="update test set id ='"+id+"', pw='"+pw+"' where seq='"+seq+"'";
sql_delete="delete from test where seq='"+seq+"'";

try
{
//mysql 드라이버의 연결
```

다음 페이지에 계속 ➊

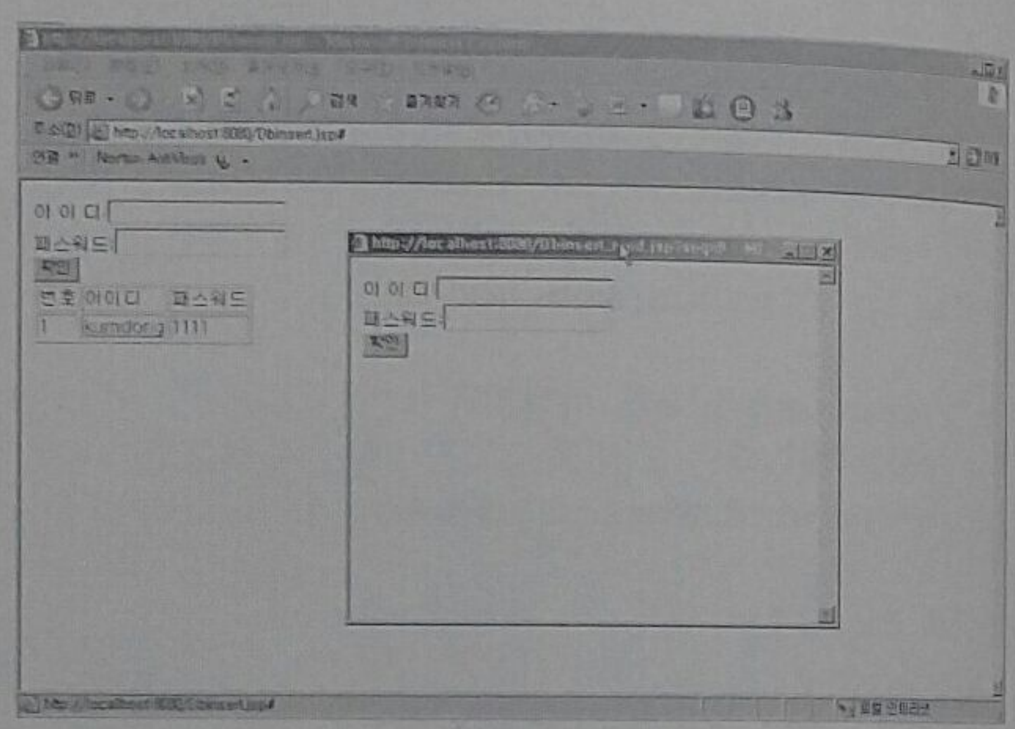
```
Class.forName("org.gjt.mm.mysql.Driver");
Connection Conn =
        DriverManager.getConnection("jdbc:mysql://localhost/testjsp","root","");
Statement stmt = Conn.createStatement();

//save_mod_del :1->저장, 2->수정, 3->삭제
if("1".equals(save_mod_del))
{
  //executeUpdate를 사용하면 리턴 값이
  //정상적일 때는 1 비정상적일 때는 -1을 리턴
  message=stmt.executeUpdate(sql_save);
}
else if("2".equals(save_mod_del))
{
  //executeUpdate를 사용하면 리턴 값이
  //정상적일 때는 1 비정상적일 때는 -1을 리턴
  message=stmt.executeUpdate(sql_update);
}
else if("3".equals(save_mod_del))
{
  //executeUpdate를 사용하면 리턴 값이
  //정상적일 때는 1 비정상적일 때는 -1을 리턴
  message=stmt.executeUpdate(sqsql_deletel);
}
stmt.close();
Conn.close();
}
catch(SQLException e)
{
 out.println(e);
}
catch(Exception e1)
{
 out.println(e1);
}

if(message > 0)
{
%>
<form name="frm" method="post" action="./Dbinsert.jsp">
</form>
<script language="javascript">
  window.opener.document.formm.submit();
  window.close();
</script>
<%
}
else
{
%>
<form name="frm" method="post" action="./Dbinsert.jsp">
</form>
<script language="javascript">
  window.opener.document.formm.submit();
  window.close();
</script>
<%
}
%>
```

❺ 브라우저를 열고 Dbinsert.jsp를 호출한다.



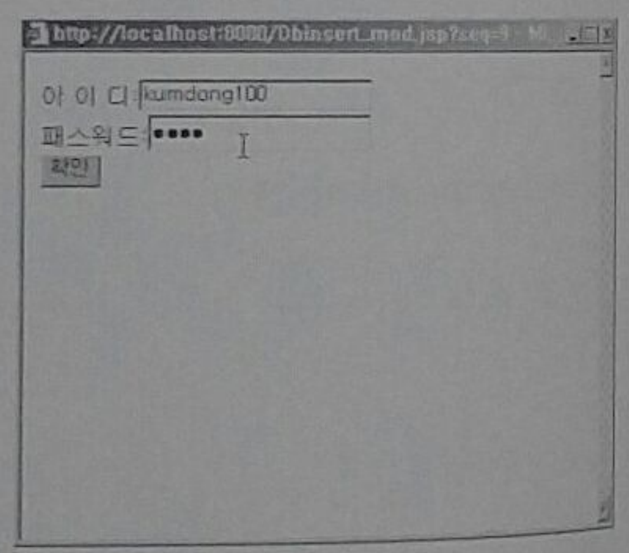
▶ Dbinsert.jsp의 호출

❻ 리스트에서 링크된 아이디를 클릭하여 수정 윈도우를 띄운다.



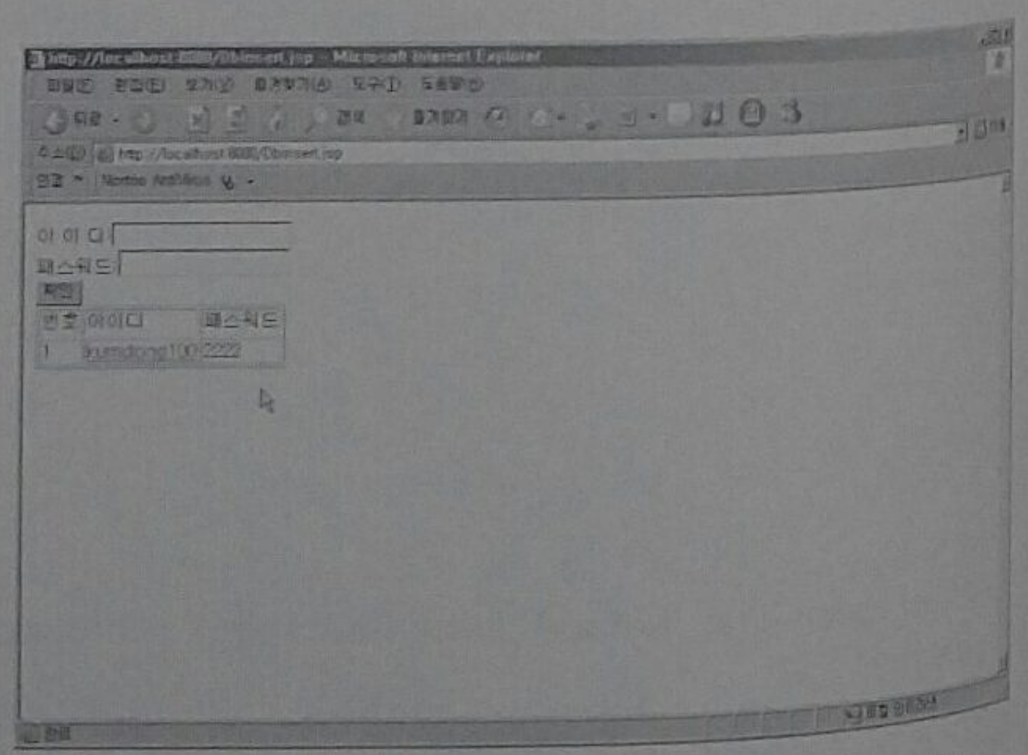
▶ 수정 윈도우

❼ 변경되는 아이디 패스워드 값을 입력한다.



▶ 변경 값의 입력

❽ 확인 버튼을 클릭하면 변경된 값이 적용된 것을 볼 수 있다.



▶ 변경확인

데이터가 올바르게 수정이 되었다면 이제 삭제에 대한 부분을 작성해 보자. 이미 수정을 위해 리스트에서 아이디를 값으로 사용하였다. 따라서 건으로 삭제하기 위해서 리스트에 삭제라는 필드를 만들어 링크를 걸도록 한다.

기본예제- 데이터의 삭제

따라하기 데이터의 삭제

❶ Dbinsert.jsp 파일에 삭제를 위한 부분을 링크를 작성한다.

```
<tr>
    <td><%=num%></td>
    <td><a href="#" onClick="javascript:go_mod('<%=rs.getString(1)%>')"><%=rs.getString(2)%></a></td>
    <td><%=rs.getString(3)%></td>
    <td><a href="#" onClick="javascript:go_del('<%=rs.getString(1)%>')">삭제</a></td>
</tr>
<%
    num++;
    }
    rs.close();
    stmt.close();
    Conn.close();


  }
  catch(SQLException e)
  {
      out.println(e);
  }
  catch(Exception el)
  {
      out.println(el);
  }
//저장된 데이터가 하나도 없을시에 출력되는 메세지
if(num ==1)
{

%>
    <tr>
        <td colspan="4">저장된 데이터가 없습니다.</td>
    </tr>
<%
}
```

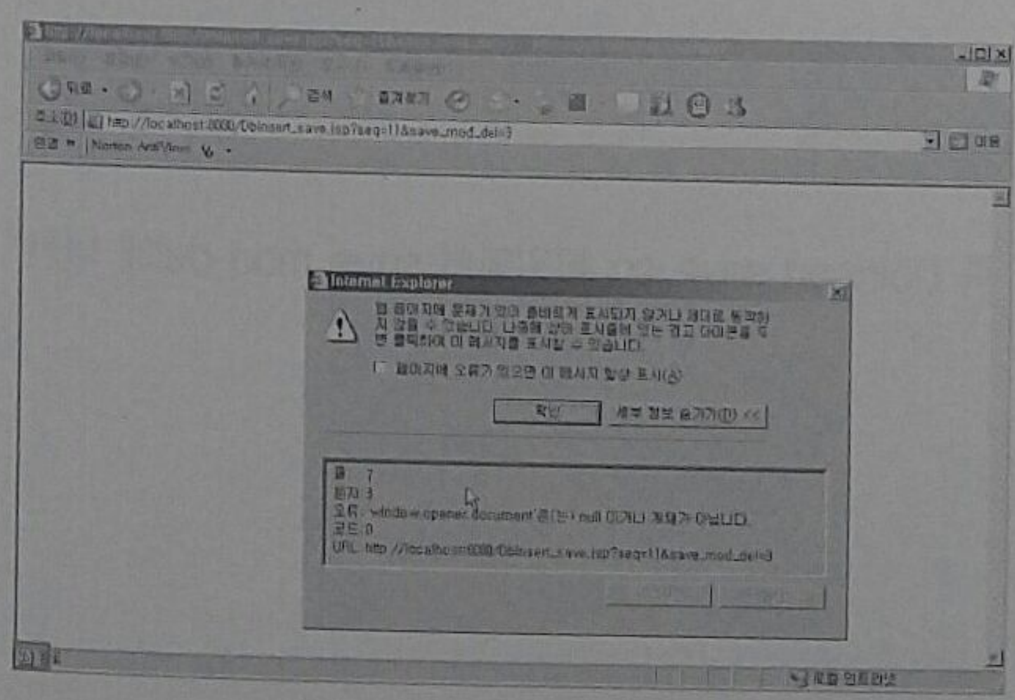
▶ 삭제를 위한 링크

❷ 삭제 스크립트를 작성한다. 우리는 여기서 Dbinsert_ save.jsp 내에서 이미 save_mod_ del의 파라메타 값이 3이면 삭제 쿼리를 실행하라고 정해 두었다. 따라서 넘겨지는 파라메타의 값 또한 그에 맞게 맞춰주어야 한다.

```
function go_del(seq)
{
    var frm = document.formm;
    if(confirm("삭제하시겠습니까?"))
    {
        frm.action="./Dbinsert_save.jsp?seq="+seq+"&save_mod_del=3";
        frm.submit();
    }
}
```

▶ 스크립트의 작성

❸ 브라우저를 열고 Dbinsert.jsp 파일을 호출하여 삭제를 클릭한다. 그럼 스크립트 에러를 발생시킨다.



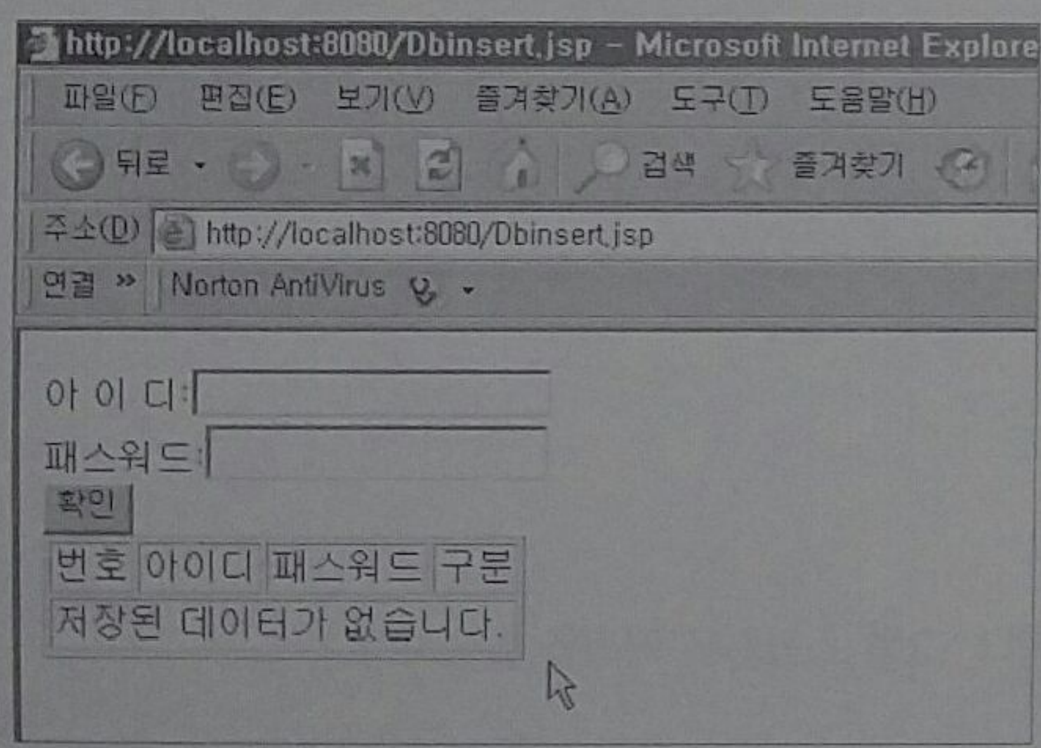
▶ 스크립트 에러

이 부분은 Dbinsert_save.jsp 파일에서 저장, 삭제, 수정을 수행하고 나서 반환되는 페이지로 이동시에 window.opener에 대한 메시지다. 즉 수정은 수정 윈도우를 이용하여 수정하였지만 저장이나 삭제는 윈도우를 이용하여 삭제 저장하는 것이 아니기 때문에 그에 따른 에러를 발생하는 것이다. 따라서 리턴되는 페이지로 이동하는 스크립트는 변경이 되어야 한다.

```
if(message > 0 && "2".equals(save_mod_del))
{
%>
<form name="frm" method="post" action="./Dbinsert.jsp">
</form>
<script language="javascript">
      window.opener.document.formm.submit();
      window.close();
</script>
<%
}
else if(message > 0 && "1".equals(save_mod_del))
{
%>
<form name="frm" method="post" action="./Dbinsert.jsp">
</form>
<script language="javascript">
      document.frm.submit();
</script>
<%
}
else if(message > 0 && "3".equals(save_mod_del))
{
%>
<form name="frm" method="post" action="./Dbinsert.jsp">
</form>
<script language="javascript">
      document.frm.submit();
</script>
<%
}
%>
```

▶ 스크립트의 변경

❹ 다시 Dbinsert.jsp를 호출하여 삭제해 보면 제대로 적용이 되는 것을 확인한다.



▶ 삭제결과

❺ 마지막으로 입력 부분에 대한 소스도 수정이 되어야 한다. 즉 Dbinsert_save.jsp 파일에서 save_mod_del에 대한 파라메타를 받아 쿼리를 실행하기 때문이다.

```
if(confirm("저장하시겠습니까?"))
{
    frm.action="./Dbinsert_save.jsp?save_mod_del=1";
    frm.submit();
}
```

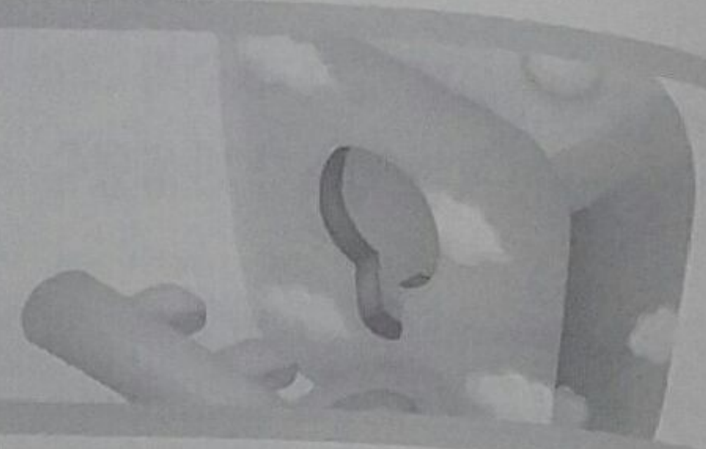
▶ 저장 스크립트의 수정

예제를 실행하면 입력과 수정 그리고 삭제가 자유자제로 작동되는 것을 확인할 수 있다. 좀더 완벽한 부분을 보안하려면 에러가 발생했을 시에 처리하는 스크립트 부분을 추가하면 된다.

맺음말

이번 장에서는 TMJ를 통해 데이터를 저장, 수정, 삭제 그리고 저장된 데이터를 리스트로 출력해 보았다. 기본 예제를 통해 독자 스스로가 다른 프로그램을 만들어 보는 것도 좋은 방법이다. 이번 장에서 MySql과의 직접적인 연결 방법을 통한 프로그램 예제였다면 다음 장에서는 앞에서 언급한 바와 같이 DBCP를 설정한 방법을 통해 TMJ를 구현해 보도록 한다.

10 DBCP를 이용한 TMJ



안녕하세요

이번 장에서는 MySql과의 직접적인 연결 방법이 아닌 톰캣에서 제공하는 Pool을 이용한 DBCP 연결 방법에 대해서 알아볼 것이다. 일반적으로 DBCP를 이용한 방법을 많이 사용함으로 독자는 설정 방법과 사용방법에 대해서 반드시 숙지하기 바란다.

DBCP가 필요한 이유

DBCP(DataBase Connection Pool)는 다중 접속자를 위한 방법이다. 쉽게 이야기 하자면 한 사이트에 여러 사용자가 접속하다 보면 서버 부하를 발생 시킨다. 그럼으로써 서버다운의 현상을 발생시키게 된다. 따라서 사용하지 않은 접속을 끊고 그 접속에 대한 권한을 다시 가져와 접속을 요청하는 접속자에게 그 권한을 재부여 해주는 것을 말한다.

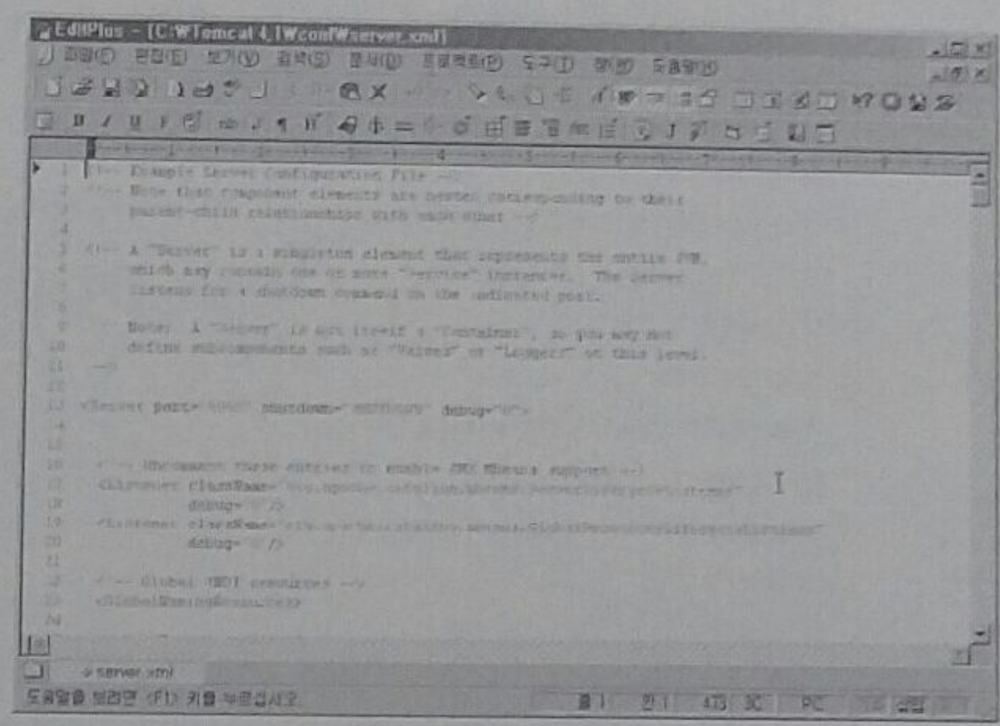
이런 과정을 통해 서버의 효율성 및 속도나 퍼포먼스를 높이기 때문이다. 예전의 톰캣의 경우에는 독립적이지 못하고 아파치와 연동하여 사용했으나 4.X 이후에는 향상된 기능으로 독립적인 구성이 가능하게 되었다.

톰캣에서의 DBCP의 설정

앞서 예제에서 프로그램의 루트가 항상 tomcat/Root에 위치해야만 했다. 하지만 앞으로는 디렉토리를 만들고 그 아래에 여러 개의 폴더와 필요한 JSP 파일 또는 JS 파일을 위치할 수 있다. 즉 사용자가 원하는 디렉토리를 지정함으로 Root가 아닌 사용자 지정 디렉토리를 Context로 지정함으로서 다양한 사이트 구성이 가능하다. 이를 위해서 톰캣에서는 Context를 만들어 주는데 Tomcat 폴더 아래에 있는 conf 폴더에 존재하는 server.xml 파일을 열고 Context를 지정해 준다.

따라하기 DBCP의 설정

❶ Tomcat 폴더의 conf 폴더를 열고 server.xml 파일을 연다.



▶ server.xml 파일

❷ server.xml 파일 내에서 <!—Tomcat Root Context —> 부분을 주석 처리한다.

```
<!-- Tomcat Root Context -->
<!--
  <Context path="" docBase="ROOT" debug="0"/>
-->

    <!-- Tomcat Examples Context -->
<Context path="/examples" docBase="examples" debug="0"
         reloadable="true" crossContext="true">
  <Logger  className="org.apache.catalina.logger.FileLogger"
             prefix="localhost_examples_log." suffix=".txt"
      timestamp="true"/>
  <Ejb   name="ejb/EmplRecord" type="Entity"
         home="com.wombat.empl.EmployeeRecordHome"
       remote="com.wombat.empl.EmployeeRecord"/>
```

▶ Root Context의 주석처리

❸ 그 아래에 다음과 같이 Context를 넣고 DBCP를 지정한다.

```
<!-- Tomcat TMJ Context -->
<Context path="/tmj" docBase="c:\tmj" debug="0" reloadable="true">
<Logger className="org.apache.catalina.logger.FileLogger"
                prefix="office_log." suffix=".txt"  timestamp="true"/>
  <Resource name="jdbc/tmjtest" auth="Container" type="javax.sql.DataSource"/>
  <ResourceParams name="jdbc/tmjtest">
    <parameter>
      <name>username</name>
      <value>root</value>
    </parameter>
    <parameter>
      <name>password</name>
      <value></value>
    </parameter>
    <parameter>
      <name>driverClassName</name>
      <value>com.mysql.jdbc.Driver</value>
    </parameter>
    <parameter>
      <name>url</name>
      <value>jdbc:mysql://localhost:3306/tmjtest?autoReconnect=true</value>
    </parameter>
    <parameter>
      <name>removeAbandoned</name>
      <value>true</value>
```

다음 페이지에 계속 ➲

```
    </parameter>
    <parameter>
      <name>removeAbandonedTimeout</name>
      <value>60</value>
    </parameter>
    <parameter>
      <name>logAbandoned</name>
      <value>true</value>
    </parameter>
    <parameter>
      <name>maxActive</name>
      <value>1000</value>
    </parameter>
    <parameter>
      <name>maxIdle</name>
      <value>10</value>
    </parameter>
    <parameter>
      <name>maxWait</name>
      <value>180</value>
    </parameter>
  </ResourceParams>
</Context>
```

TIP DBCP 설정 XML 상세설명

❶ 데이터베이스 접속 ID를 입력한다.

```
<parameter>
    <name>username</name>
    <value>root</value>
</parameter>
```

❷ 데이터베이스 접속 암호를 입력한다.(예제는 암호를 지정하지 않았음)

```
<parameter>
    <name>password</name>
    <value></value>
</parameter>
```

❸ 사용되는 JDBC 드라이버를 명시한다.

```
<parameter>
    <name>driverClassName</name>
    <value>com.mysql.jdbc.Driver</value>
</parameter>
```

❹ JDBC 연결 URL로 DB 서버와 네트워크가 끊겼을 경우 POOL이 자동으로 Connection 들을 재설정해 준다.

```
<parameter>
    <name>url</name>
    <value>jdbc:mysql://localhost:3306/tmjtest?autoReconnect=true</value>
</parameter>
```

다음 페이지에 계속 ❍

TIP

❺ 사용가능한 DB 연결이 적을 경우 DBCP가 활용되지 않는 또는 버려진 연결을 찾아 재활용하게 된다.

```
<parameter>
    <name>removeAbandoned</name>
    <value>true</value>
</parameter>
```

❻ DBCP에서 버려진 접속으로 인식할 때까지의 시간을 지정(초단위로 지정)할 수 있다. 기본이 60 = 5분=300초이다.

```
<parameter>
    <name>removeAbandonedTimeout</name>
    <value>60</value>
</parameter>
```

❼ 매개변수를 true로 설정하면 사용되지 않는 DB 연결을 유발하는 코드를 찾을 수 있는 로그를 작성하게 해 준다. 기본값은 false이다.

```
<parameter>
    <name>logAbandoned</name>
    <value>true</value>
</parameter>
```

❽ DB 연결 풀의 최대 연결 개수로 0을 입력하면 무제한이며 숫자를 지정하면 MySql의 데몬의 최대 연결 개수에 따라서 그 값이 변동된다. 2CUP/2G Mem 경우 1000정도면 적당하다고 본다.

```
<parameter>
    <name>maxActive</name>
    <value>1000</value>
</parameter>
```

❾ DB 연결 풀에서 보관되는 최대 수로 0은 무제한이다.

```
<parameter>
    <name>maxIdle</name>
    <value>30</value>
</parameter>
```

❿ DB 연결이 반환되는 시간으로 이 시간을 초과하는 경우 예외가 발생이 된다. 1/10000이 되며 -1은 무한대기이다(-1은 사용하지 말 것).

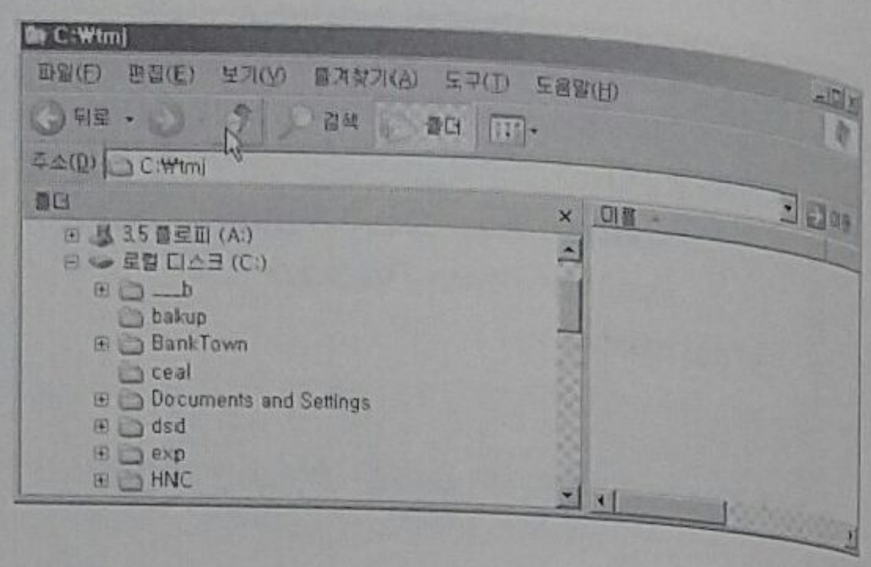
```
<parameter>
    <name>maxWait</name>
    <value>180</value>
</parameter>
```

❹ Conf 폴더에서 web.xml 파일을 열고 맨 마지막 줄에 server.xml 파일에서 지정한 데이터베이스 이름을 지정한다.

```
    <welcome-file-list>
        <welcome-file>index.html</welcome-file>
        <welcome-file>index.htm</welcome-file>
        <welcome-file>index.jsp</welcome-file>
    </welcome-file-list>

    <resource-ref>
        <description>MySql Datasource</description>
        <res-ref-name>jdbc/tmjtest</res-ref-name>
        <res-type>javax.sql.DataSource</res-type>
        <res-auth>Container</res-auth>
    </resource-ref>

</web-app>
```

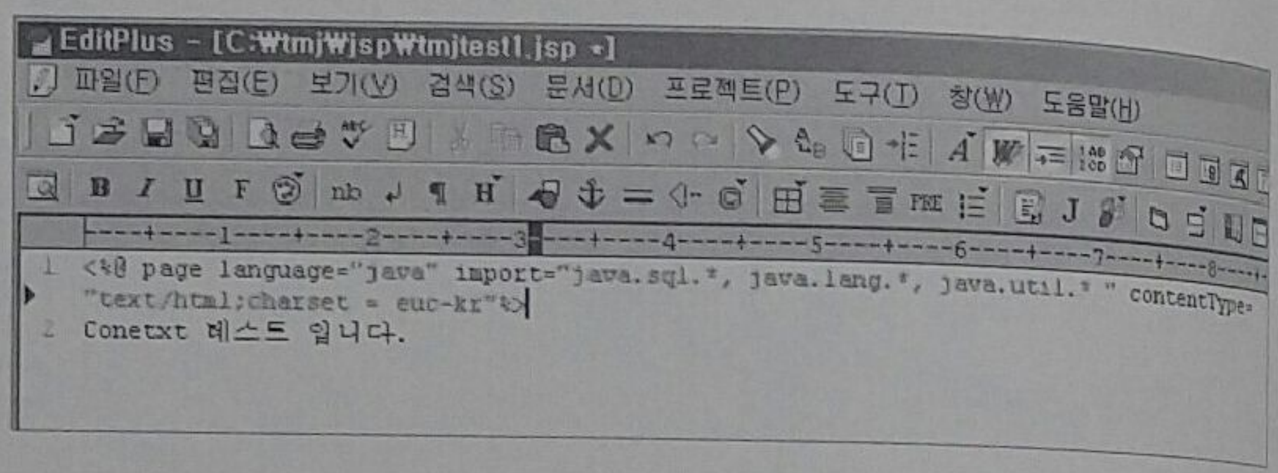
▶ web.xml의 추가

❺ tmj라는 폴더를 생성한다.



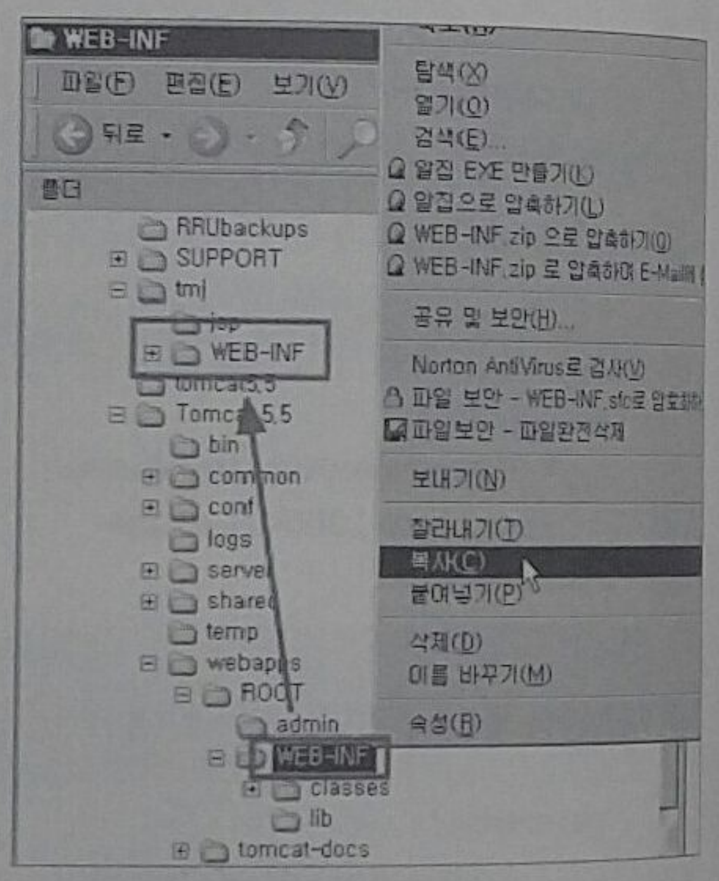
▶ Context 폴더 tmj의 생성

❻ Context가 제대로 되었는지 간단한 JSP 파일을 만들어 tmjtest1.jsp로 저장한다.



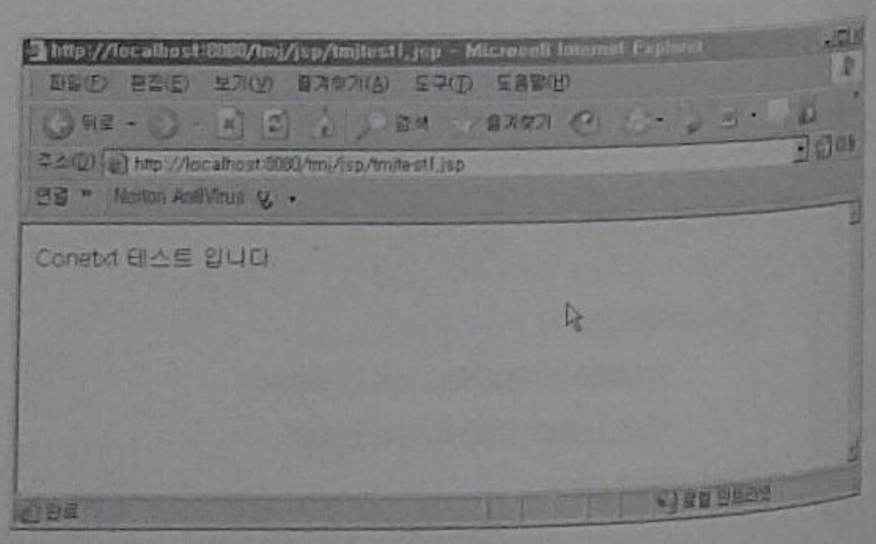
▶ tmjtest1.jsp

❼ Class 파일을 위해 Tomcat에 있는 WEB-INF의 폴더를 복사하여 tmj 폴더 안에 붙여넣기를 한다.



▶ WEB-INF의 복사와 붙여 넣기

❽ 브라우저를 열고 Context로 지정한 경로의 tmjtest1.jsp 파일을 호출한다.



▶ tmjtest1.jsp의 호출

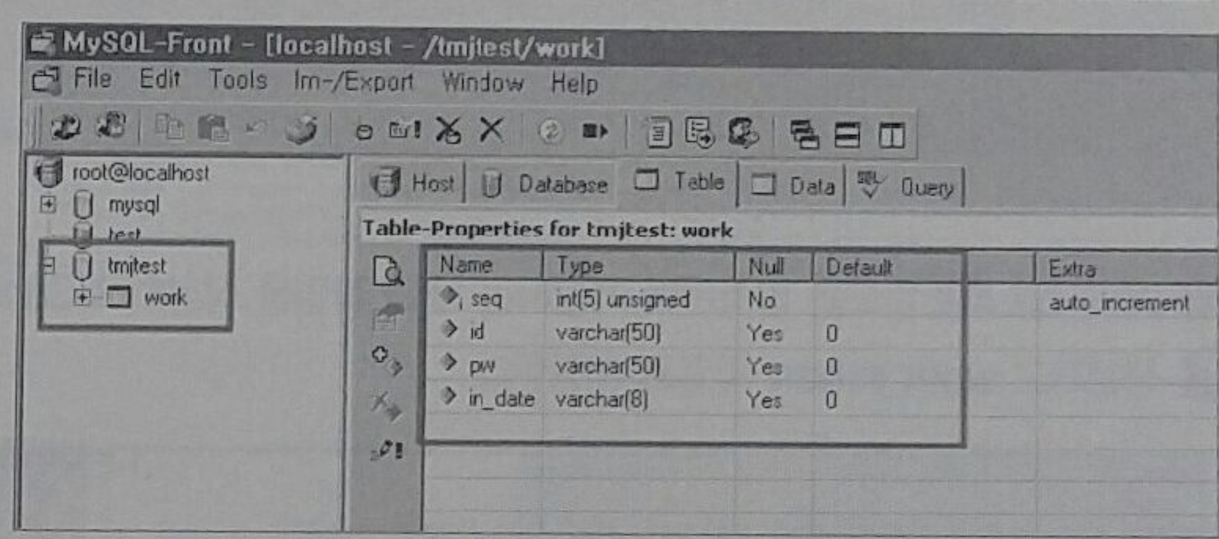
이로써 Context가 설정이 되었다. 앞으로는 생성된 Context 폴더 안에 jsp, html, js, include 등의 폴더를 만들어 해당 되는 파일을 저장하면 된다. Context를 만들면 나중에 서버가 이전되더라도 Context 내에 있는 폴더만 복사해서 server.xml 파일에 Context만 지정해 주면 운영체계에 상관없이 구현된다.

기본예제 - DBCP를 이용한 입력 값의 저장

앞 예제는 직접적인 MySql과의 연결이었다. 프로그램 상에서 직접 코딩을 하여 MySql에 접속하는 프로그램을 작성할 때마다 코딩을 해 주었다. 이러한 불편한 점을 없애기 위해 MySql에 접속하기 위한 프로그램을 작성하고 단지 JSP 내에서 include만 시키면 된다.

따라하기 DBCP를 이용한 입력 값의 저장

❶ 먼저 데이터베이스인 MySql에 Tomcat의 server.xml에서 설정한 JDBC에 해당하는 데이터베이스를 생성하고 그 안에 사용자 정보를 담는 테이블을 생성한다.



▶ tmjtest 데이터베이스의 생성

❷ 접속하는 프로그램을 작성하고 Exception 처리된 데이터베이스 접속 프로그램을 작성하여 tmj에 하위폴더 include 폴더를 생성하여 try.jsp 파일로 저장한다.

```
<%
Connection con=null;
Statement stmt=null;
ResultSet rs =null;

try {
    Context initCtx = new InitialContext();
    Context envCtx = (Context) initCtx.lookup("java:comp/env");
    DataSource ds = (DataSource) envCtx.lookup("jdbc/tmjtest");

    con = ds.getConnection();
    stmt = con.createStatement();
%>
```

try.jsp 프로그램에서 눈여겨 봐둘 부분은 다음이다.

```
DataSource ds = (DataSource) envCtx.lookup("jdbc/tmjtest");
```

DBCP에서 지정된 이름을 네이밍 서비스에 같은 이름을 부여하고 같은 이름의 데이터베이스가 있다는 것이다. 만약 이름이 같지 않는다면 데이터베이스에 접속할 수 없다는 오류 메시지가 출력된다.

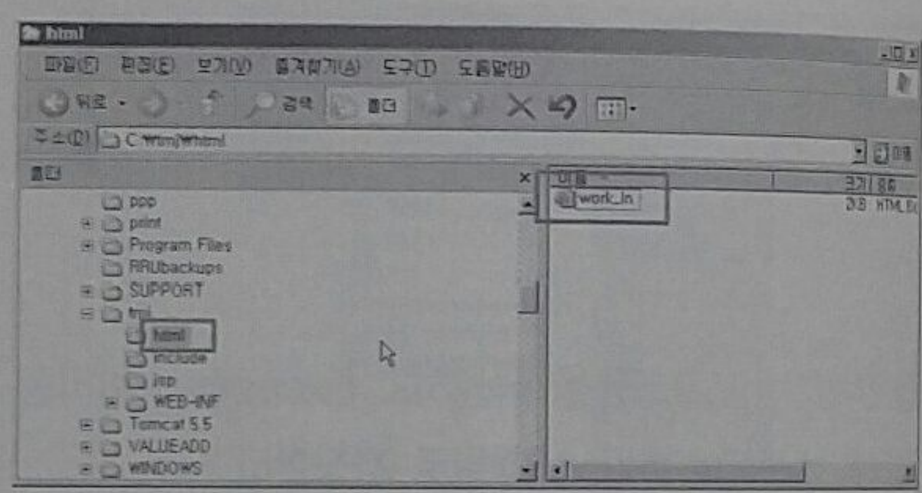
❸ 다음은 try문과 항상 짝을 이루는 catch문이 들어가 있는 프로그램을 작성하고 tmj에 하위폴더 include 폴더를 생성하여 catch.jsp 파일로 저장한다.

```
<%
    }catch(NamingException ne){
        out.println(ne.toString());
    }catch(SQLException se){
        out.println(se.toString());
    }finally{
        try { if (rs != null) rs.close(); } catch (Exception e) {}
        try { if (stmt != null) stmt.close(); } catch (Exception e) {}
        try { if (con != null) con.close(); con = null;} catch (Exception e) {}
    }
%>
```

❹ 아이디와 패스워드를 입력하여 저장하는 html 파일을 작성하고 Context로 지정된 tmj 폴더에 html이라는 하위폴더를 생성하고 work_in.html 파일로 저장한다.

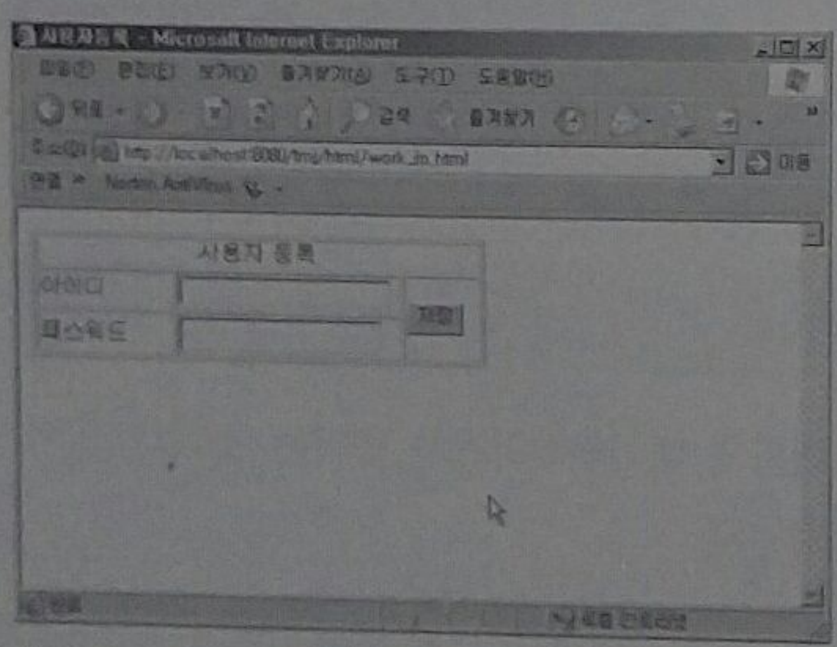
```
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=euc-kr">
<title>사용자등록</title>
</head>
<body bgcolor="white" text="black" link="blue" vlink="purple" alink="red">
<form name="workInForm" method="post">
<table border="1" width="324" height="75">
    <tr>
        <td width="314" colspan="3" height="20">
            <p align="center">사용자 등록</p>
        </td>
    </tr>
    <tr>
        <td width="95">아이디</td>
        <td width="156"><input type="text" name="workId"></td>
        <td width="50" height="43" rowspan="2">
                <p><input type="button" name="save_btn" value="저장" ▷
</td>
    </tr>
    <tr>
        <td width="95" height="11">패스워드</td>
        <td width="156" height="23"><input type="password" name="workPw"></td>
    </tr>
</table>
</form>
</body>
</html>
```

▶ work_in.html의 작성



▶ tmj/html/에 저장

❺ 브라우저를 열고 work_in.html을 호출해 본다.



▶ work_in.html의 호출

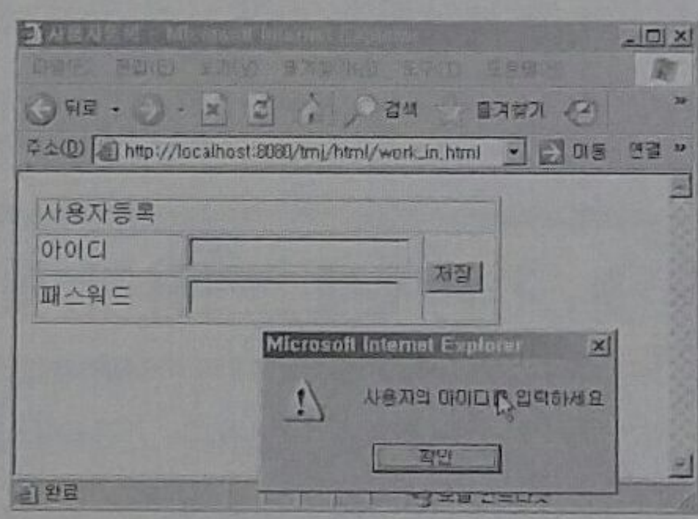
❻ 작성된 work_in.html 파일을 열고 앞서서 배운 예제를 기억하여 입력 값을 체크하는 스크립트를 작성한다.

```
<script language="javascript">
    //입력값을 체크하는 function
    function go_save()
    {
        var TheForm = document.workInForm;
        if(TheForm.workId.value =='')
        {
            alert('사용자의 아이디를 입력하세요');
        }
        else if(TheForm.workPw.value =='')
        {
            alert('사용자의 패스워드를 입력하세요');
        }
        else
        {
            TheForm.action="../jsp/workSave.jsp";
            TheForm.submit();
        }
    }
</script>
</head>
<body bgcolor="white" text="black" link="blue" vlink="purple" alink="red">
<form name="workInForm" method="post">
<table border="1" width="324" height="75">
    <tr>
        <td width="314" colspan="3" height="20">
            <p align="center">사용자 등록</p>
        </td>
    </tr>
    <tr>
        <td width="95">아이디</td>
        <td width="156"><input type="text" name="workId"></td>
        <td width="50" height="43" rowspan="2">
            <p><input type="button" name="save_btn" value="저장" onClick="javascript:go_save();">
</td>
    </tr>
```

저장루틴이 작성된 jsp

▶ 입력체크를 위한 스크립트의 작성

❼ 브라우저를 열고 입력에 대한 체크가 되는지 확인한다.



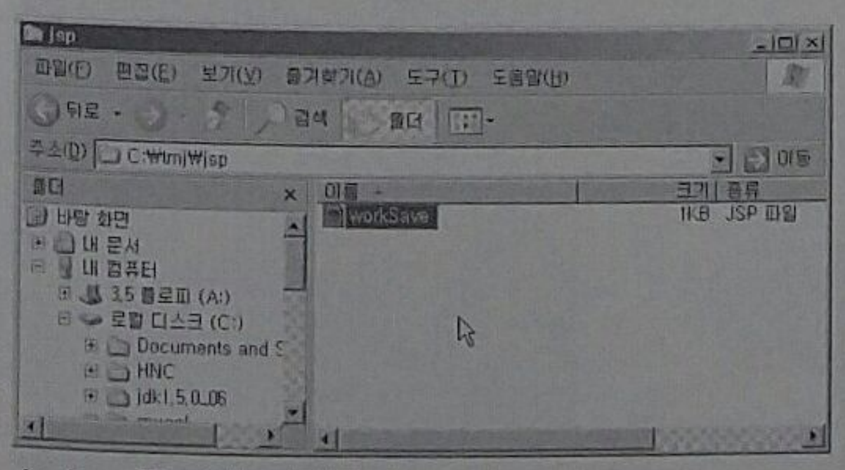
▶ 입력체크 여부 확인

❽ html이 완성이 되었다면 입력 값을 받아 MySql에 저장하는 프로그램을 작성한다.

```
<%@ page import="java.net.*,java.io.*,java.sql.*,java.util.*,javax.naming.*,javax.sql.DataSource" %>
<%
  //입력되는 아이디와 패스워드를 받는다.
  String workId = request.getParameter("workId");
  String workPw = request.getParameter("workPw");

  //입력쿼리
  //DATE_FORMAT(sysdate(), '%Y%m%d')은 서버의 년월일로 입력
String workIn_sql="insert into worker(id,pw,in_date) values('"+workId+"','"+workPw+"',DATE_FORMAT(sysdate(), '%Y%m%d'))";

%>
<%@ include file="../include/try.jsp" %>
<%
     stmt.executeUpdate(workIn_sql);
%>
<%@ include file="../include/catch.jsp" %>
```

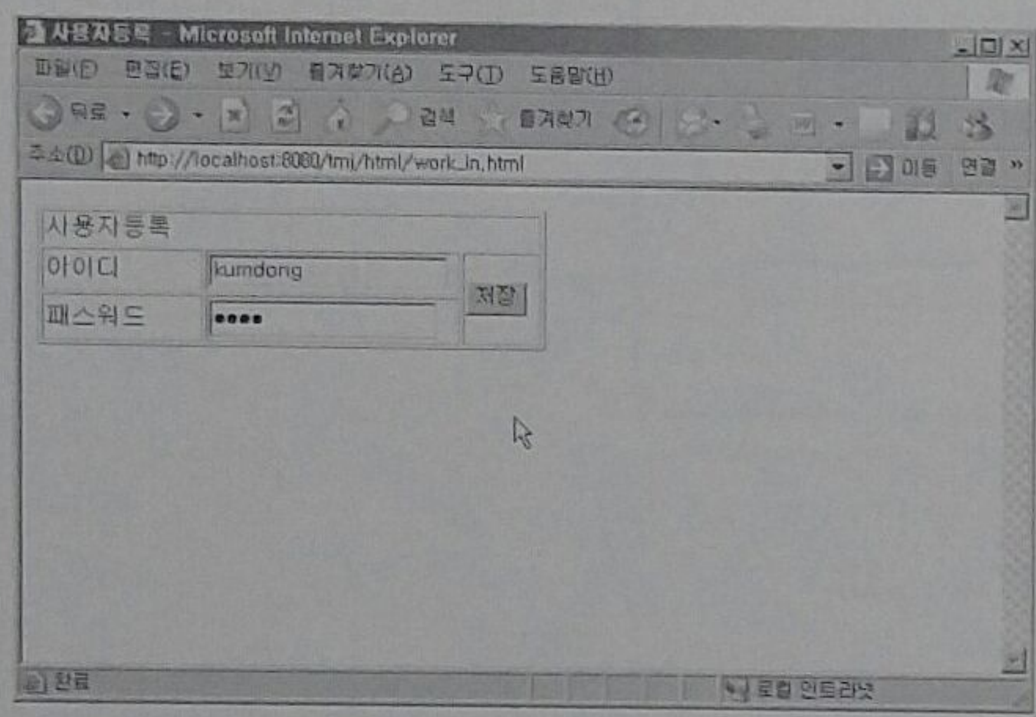
▶ workSave.jsp의 작성

❾ tmj 폴더에 jsp 폴더를 만들어 저장한다.

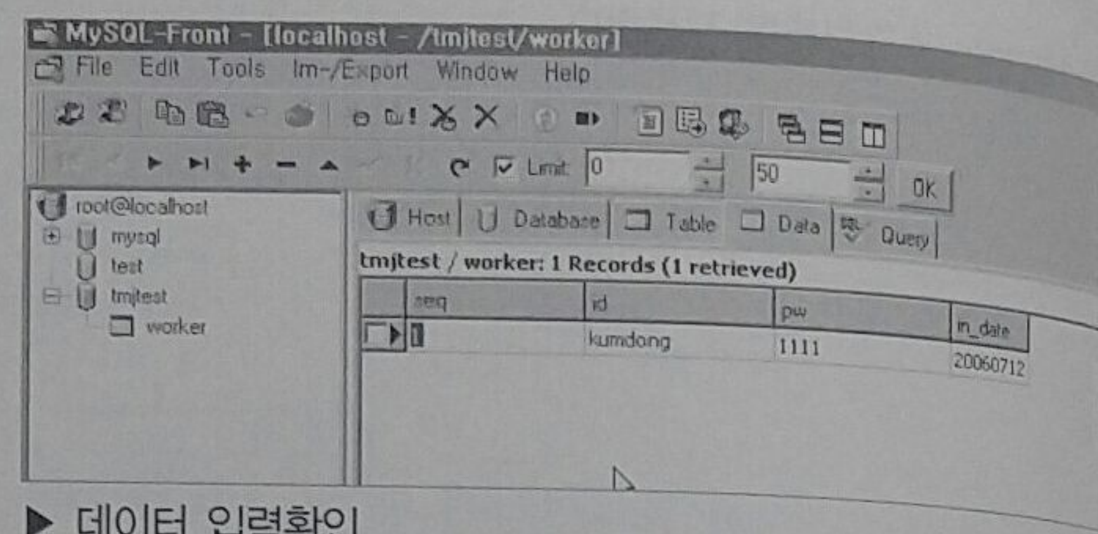


▶ jsp 폴더에 저장하기

⑩ 브라우저에서 work_in.html를 호출하여 값을 입력한 후 MySql에 만들어진 worker라는 테이블에 데이터를 확인해 본다.



▶ 브라우저에서 값 입력



▶ 데이터 입력확인

예제를 통해서 Tomcat에서 DBCP를 이용하여 데이터에 저장해 보았다. 이 예제를 더욱 발전시켜서 사용자를 체크하고 사용자가 등록이 되어 있지 않으면 등록하라는 메시지를 보여주도록 한다.

기본예제 –사용자 체크

기존의 work_in.html은 JSP에서 보내는 리턴 값을 받지 못하므로 앞서 배운 예제처럼 jsp 파일로 만든 후 아이디와 패스워드를 입력하면 기존에 등록되어 있는 사용자라면 다른 페이지로 이동하고 그렇지 않으면 사용자등록 페이지로 이동시킨다.

따라하기 사용자 체크를 위한 로그인 작성

❶ 기존의 work_in.html 파일을 메시지를 처리할 수 있는 Javascript를 만들고 work_login.jsp로 jsp 폴더에 저장한다.

```
[work_login.jsp]
<%@ page language="java" import="java.sql.*, java.lang.*, java.util.* "
                                contentType= "text/html;charset = euc-kr"%>
<%
        String msg=request.getParameter("msg");
%>
<html>
<head>
<meta http-equiv="content-type" content="text/html; charset=euc-kr">
<title>사용자등록</title>
<script language="javascript">
   function go_check()
   {
                var TheForm=document.workInFrom;
                if(TheForm.workId.value == "")
                {
                        alert('사용자의 아이디를 입력하세요.');
                }
```

다음 페이지에 계속

```
            else if(TheForm.workPw.value == "")
            {
                    alert('사용자의 패스워드를 입력하세요.');
            }
            else
            {
                //사용자 체크로 이동
                    TheForm.action="../jsp/work_check.jsp";
                    TheForm.submit();
                }
        }
//메시지를 받아 처리하는 function
        function message(msg)
        {
          var TheForm=document.workInForm;
         if(msg ='1')
                {
                        alert('아이디가 다릅니다.');
                }
               else if(msg='2')
                {
                        alert('패스워드가 다릅니다.');
                }
          else if(msg ='3')
               {
                      if(confirm('등록된 사용자가 아닙니다. 사용자로 등록하시겠습니까?'))
                      {
                            document.location.href="../html/work_in.html";

                      }
               }
        }
</script>
</head>
<body onload="javascript:message('<%=msg%>');">
<form name="workInFrom" method="post">
<table border="1" width="322">
    <tr>
        <td width="312" colspan="3">사용자로그인</td>
    </tr>
    <tr>
        <td width="97">아이디</td>
        <td width="156"><input type="text" name="workId"></td>
        <td width="46" rowspan="2">
        <input type="button" name="save_btn"
                value="로그인" onClick="javascript:go_check()">
        </td>
    </tr>
    <tr>
        <td width="97">패스워드</td>
        <td width="156"><input type="password" name="workPw"></td>
    </tr>
</table>
</form>
</body>
</html>
```

❷ work_login.jsp에서 입력받은 아이디와 패스워드를 체크하여 등록자이면 등록자 이동 페이지로 보내주고 그렇지 않으면 사용자 등록으로 이동시켜주는 work_check.jsp 파일을 작성한다.

```
[work_check.jsp]
<%@ page import="java.net.*,java.io.*,java.sql.*,java.util.*,
                                javax.naming.*,javax.sql.DataSource" %>
<%
  //입력되는 아이디와 패스워드를 받는다.
  String workId = request.getParameter("workId");
  String workPw = request.getParameter("workPw");

//체크 쿼리
String checksql_id="select count(*) from worker where id='"+workId+"'";
String checksql_pw="select count(*) from worker where pw='"+workPw+"'";

//메세지 처리를 위한 변수
int msg=0;

//아이디와 패스워드를 체크하기 위한 변수
int id_check=0;
int pw_check=0;
%>
<%@ include file="../include/try.jsp" %>
<%
     rs =stmt.executeQuery(checksql_id);
          if(rs.next())
          {
                id_check=rs.getInt(1);
          }
          rs.close();

            rs =stmt.executeQuery(checksql_pw);
                if(rs.next())
                {
                        pw_check=rs.getInt(1);
                }
                rs.close();
%>
<%@ include file="../include/catch.jsp" %>
<%
//id_check가 0이면 아이디가 없음. 1이면 아이디 존재
//pw_check가 0이면 패스워드가 없음. 1이면 패스워드 존재
//id_check,pw_check가 0보다 크면 사용자 OK
  if(id_check > 0 && pw_check > 0)
  {
%>
<form name="resultForm" method="post">
<input type="hidden" name="workNm" value="<%=workId%>">
</form>
<script language="javascript">
    document.resultForm.action="./workSucess.jsp";
    document.resultForm.submit();
</script>
<%
  }
```

다음 페이지에 계속 ➊

```
  else if(id_check == 0 && pw_check != 0)
  {
%>
<form name="resultForm" method="post">
<input type="hidden" name="msg" value="1">
</form>
<script language="javascript">
    document.resultForm.action="./work_login.jsp";
    document.resultForm.submit();
</script>
<%
  }
  else if(id_check != 0 && pw_check == 0)
  {
%>
<form name="resultForm" method="post">
<input type="hidden" name="msg" value="2">
</form>
<script language="javascript">
    document.resultForm.action="./work_login.jsp";
    document.resultForm.submit();
</script>
<%
  }
  else if(id_check == 0 && pw_check == 0)
  {
%>
<form name="resultForm" method="post">
<input type="hidden" name="msg" value="3">
</form>
<script language="javascript">
    document.resultForm.action="./work_login.jsp";
    document.resultForm.submit();
</script>
<%
  }
%>
```

❸ 앞 예제 workSave.jsp에서 저장이 완료가 되면 로그인으로 페이지로 이동할 수 있도록 수정한다.

```
[workSave.jsp]
<%@ page import="java.net.*,java.io.*,java.sql.*,java.util.*,
                                javax.naming.*,javax.sql.DataSource" %>
<%
  //입력되는 아이디와 패스워드를 받는다.
  String workId = request.getParameter("workId");
  String workPw = request.getParameter("workPw");

  //입력 쿼리
  //DATE_FORMAT(sysdate(), '%Y%m%d')은 서버의 년월일로 입력
String workIn_sql="insert into worker(id,pw,in_date) ";
 workIn_sql+="values('"+workId+"','"+workPw+"', ";
 workIn_sql+="DATE_FORMAT(sysdate(), '%Y%m%d'))";

//저장 리턴을 받기위한 변수
```

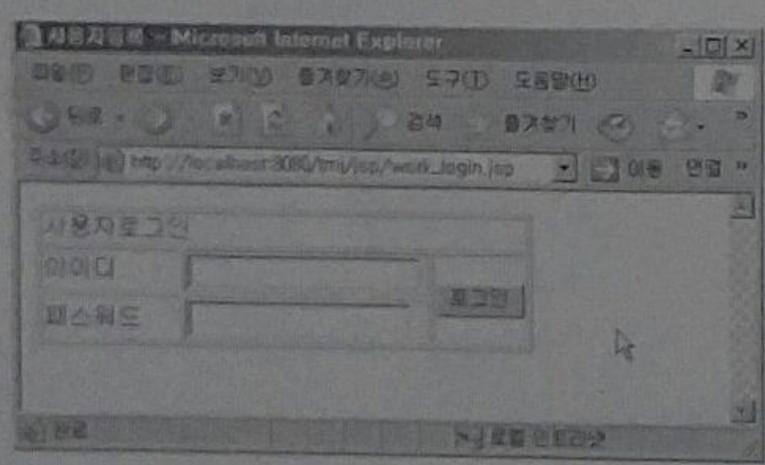
다음 페이지에 계속

```
int msg=0;
%>
<%@ include file="../include/try.jsp" %>
<%
        //정상적인 저장이면 msg에는 0보다 큰 값이 저장
        //오류일 경우 0보다 적은 값이 저장
         msg=stmt.executeUpdate(workIn_sql);
%>
<%@ include file="../include/catch.jsp" %>
<%
  if(msg > 0 )
  {
%>
<form name="resultForm" method="post">
</form>
<script language="javascript">
    document.resultForm.action="./work_login.jsp";
    document.resultForm.submit();
</script>
<%
  }
else
{
%>
<form name="resultForm" method="post">
</form>
<script language="javascript">
    document.resultForm.action="../html/work_in.html";
    document.resultForm.submit();
</script>
<%
  }
%>
```

❹ 로그인이 성공되면 메시지를 뿌려주는 간단한 페이지를 작성한다.

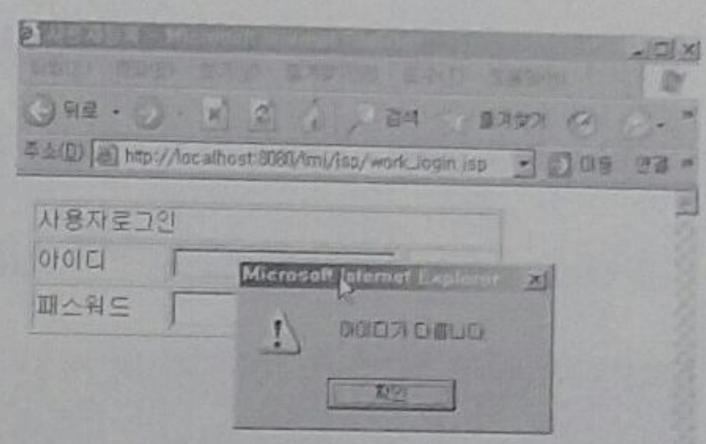
```
[workSucess.jsp]
<%@ page language="java" import="java.sql.*, java.lang.*, java.util.* "
          contentType= "text/html;charset = euc-kr"%>
<%
String workNm=request.getParameter("workNm");
%>
<%=workNm%>님이 로그인 하셨습니다.
```

❺ 브라우저를 열고 work_login.jsp를 호출한다.



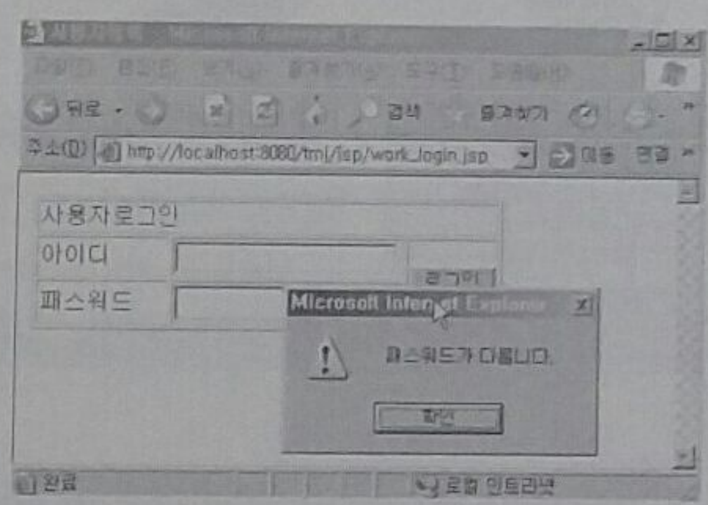
▶ work_login 호출

❻ 아이디 입력 오류 시 메시지를 처리한다.



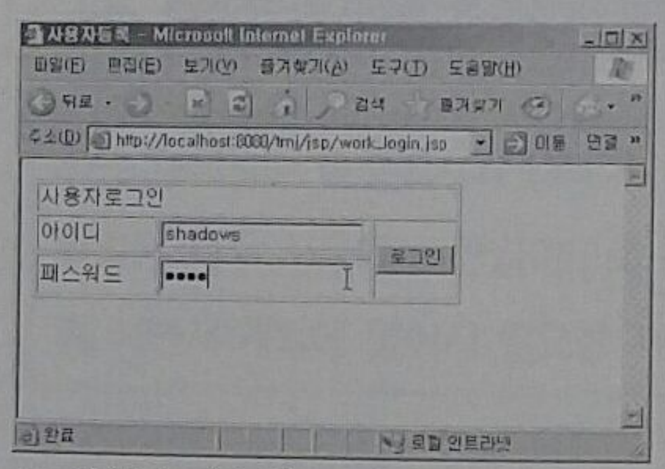
▶ 아이디 입력오류

❼ 패스워드 입력 오류 시 메시지를 처리한다.

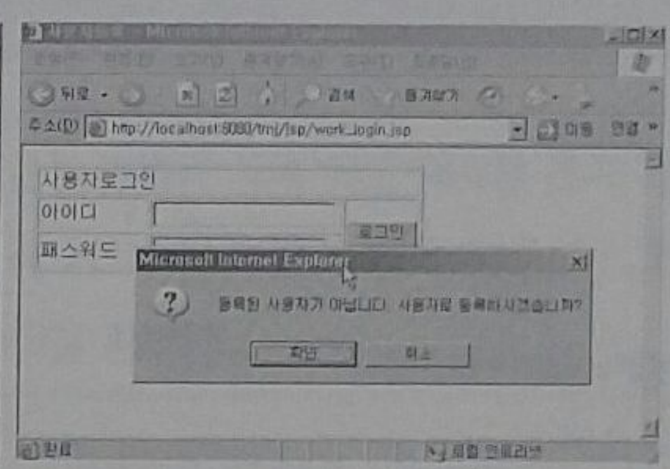


▶ 패스워드 입력오류

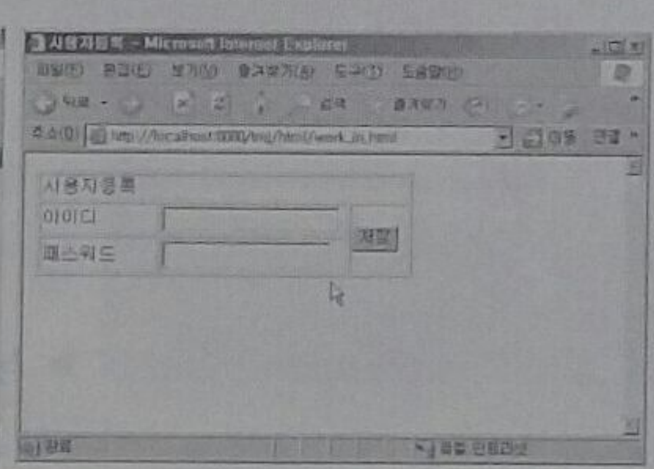
❽ 등록되지 않는 아이디와 패스워드 입력 시 로그인으로 이동한다.



▶ 아이디 패스워드 입력

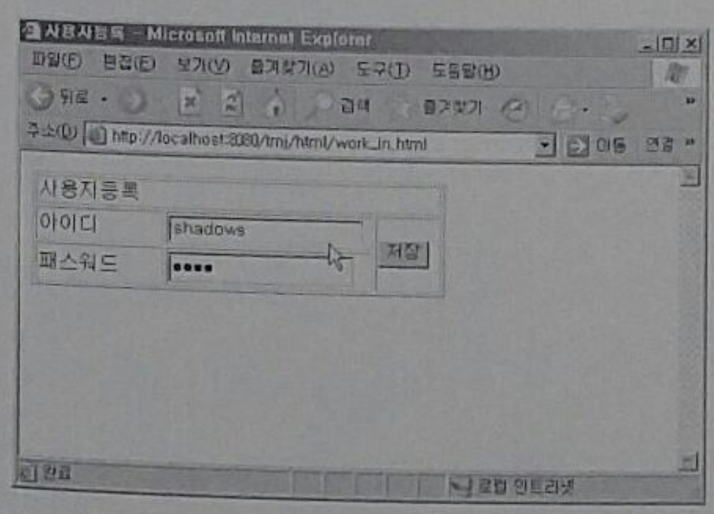


▶ 등록화면으로 이동 메시지

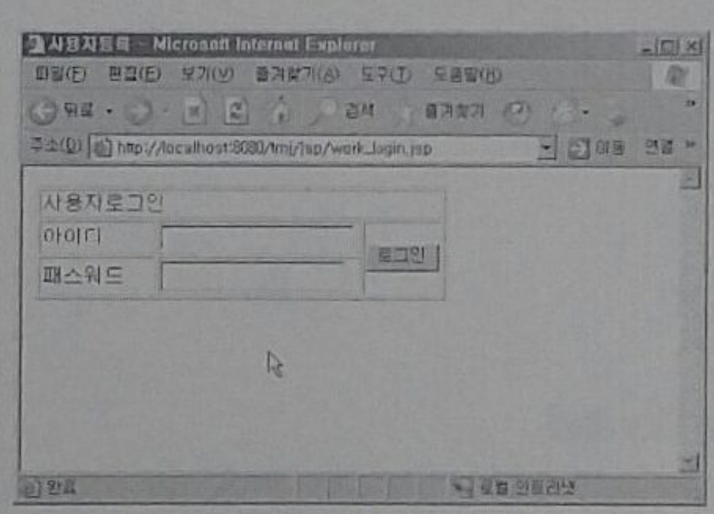


▶ 사용자등록 화면

❾ 사용자를 등록하고 저장하면 로그인 화면으로 이동된다.

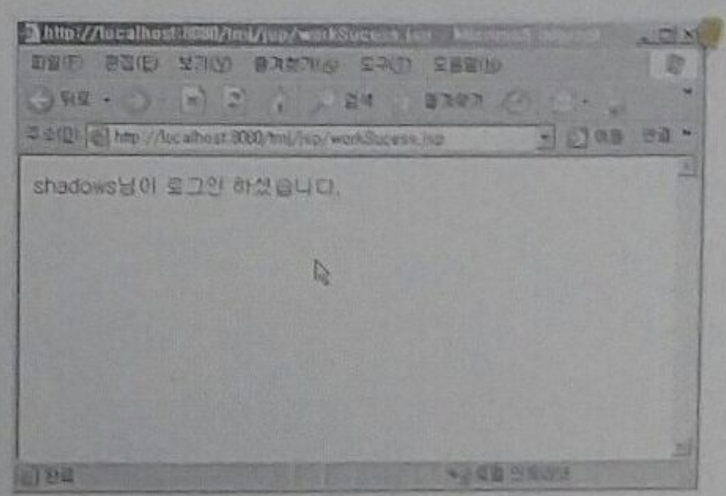


▶ 사용자등록



▶ 등록 후 로그인 화면

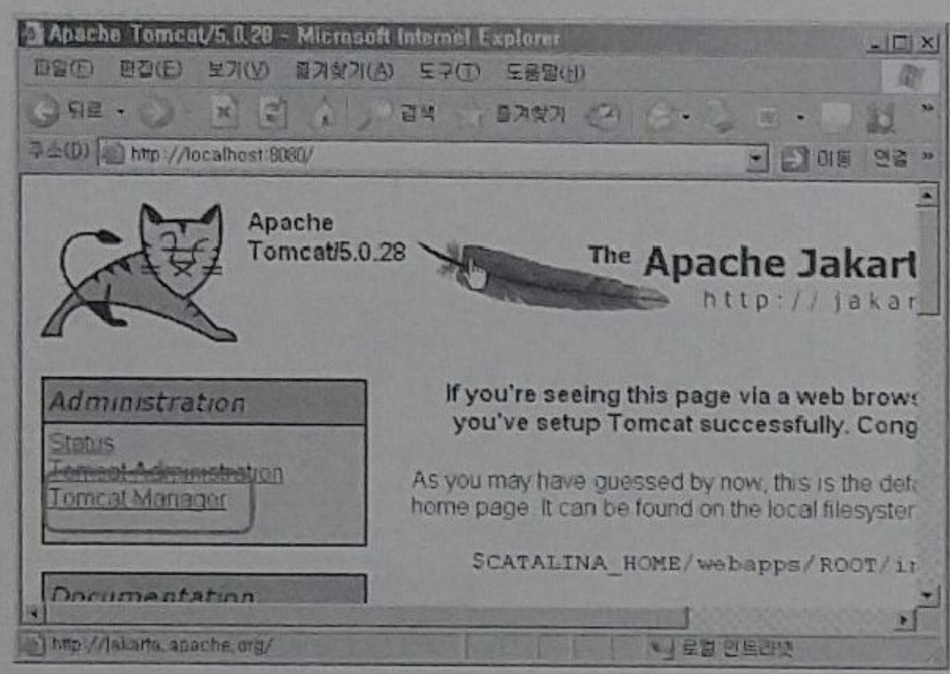
❿ 사용자 로그인을 하고 정상적인 로그인 시 아이디를 출력한다.



▶ 로그인 성공

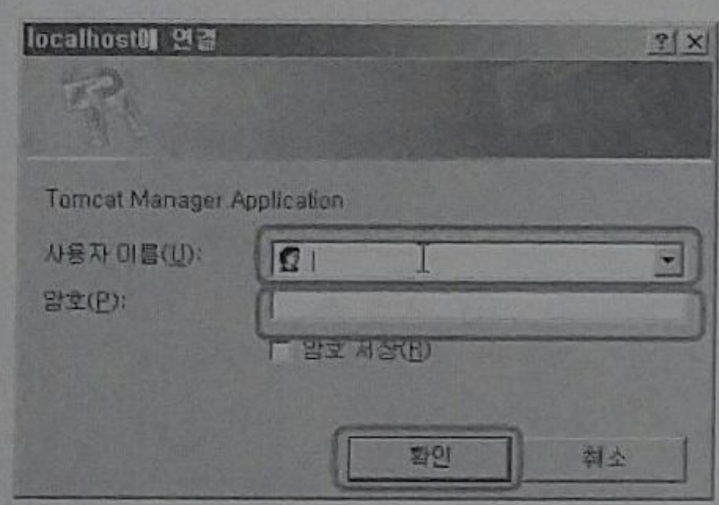
DBCP를 사용하면 Tomcat 하나에 여러 개의 작업이 가능하며 Tomcat Manager에서는 생성된 Context에 대해 개별적으로 start/ stop / restart가 가능하여 관리가 편리해 진다.

localhost:8080으로 접속하면 Administration 메뉴에 manager 항목이 있음을 볼 수 있다.



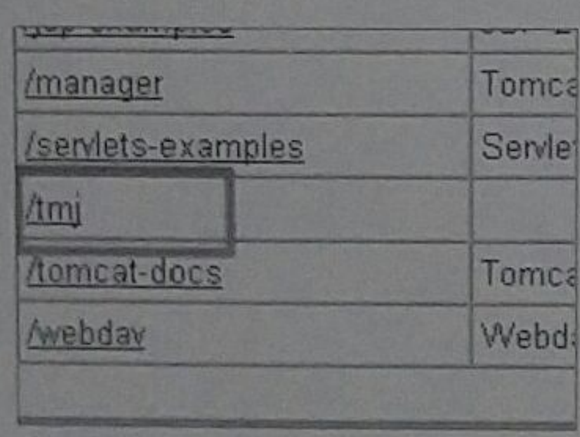
▶ Manager 항목

클릭하면 관리자의 아이디와 암호를 입력받는 윈도우가 나타난다.



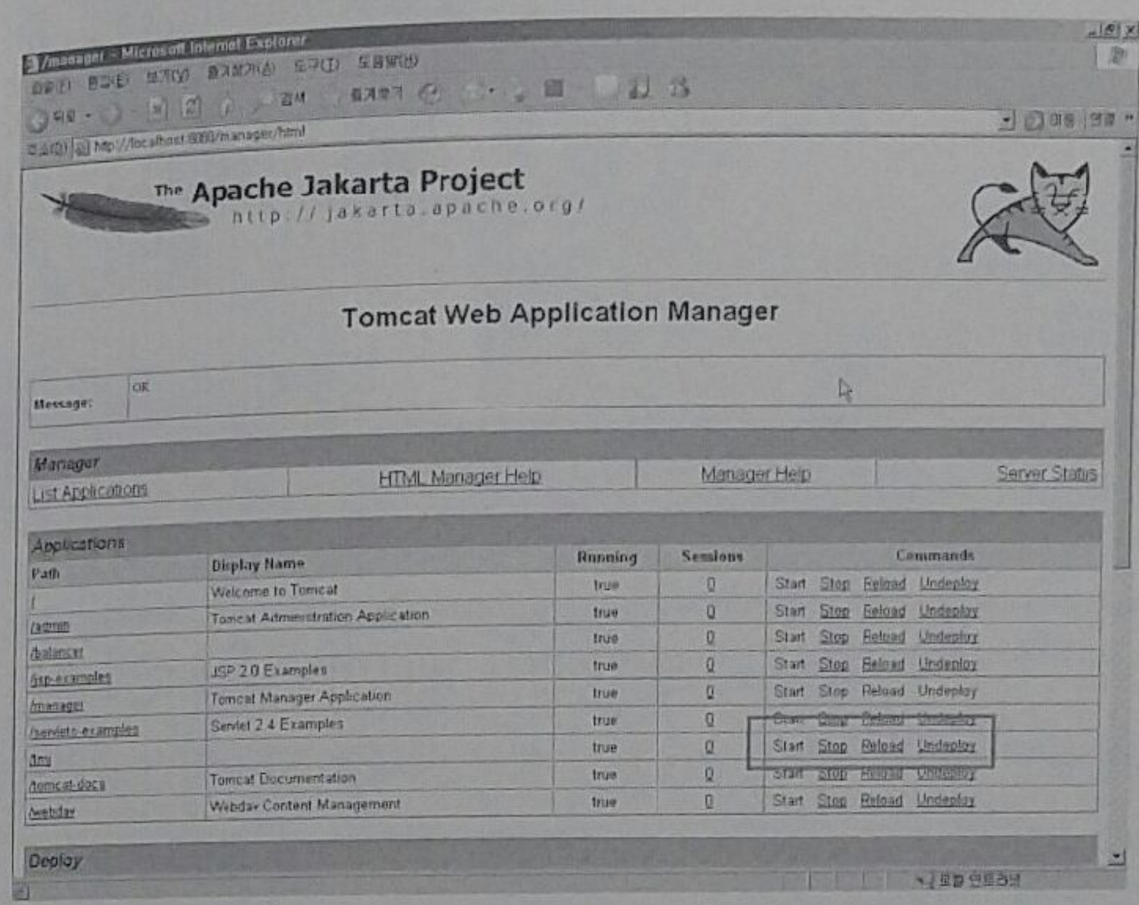
▶ Administration 접속

처음 Tomcat을 설치했을 때를 지정한 아이디와 패스워드를 입력하면 된다. 설치 시 디폴트로 설치하였기 때문에 admin이라는 아이디와 암호는 주어지지 않았다. 아이디를 입력하고 접속하면 위에서 예제 때 만든 tmj라는 Context가 존재하는 것을 볼 수 있다.



▶ tmj Context

이렇게 만들어진 Context는 개별적으로 Context 별로 start/ stop 등을 할 수 있다.



▶ commands

맺음말

이번 장에서는 톰캣에서의 DBCP 설정 방법과 그에 따른 예제를 통해 TMJ에 한발 더 다가섰다. DBCP를 이용한 방법은 자주 사용되는 편임이므로 꼭 기억하기 바란다. 다음 장에서는 그동안 배운 간단한 예제를 한층 더 발전시켜 가면서 개발에 있어서 필요한 사항을 공부한다.