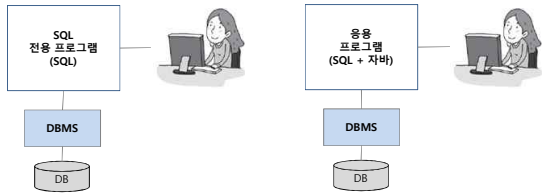
01. 데이터베이스 프로그래밍의 개념

- **'프로그래밍'이란?**
  - 프로그램을 설계하고 소스코드를 작성하여 디버깅하는 과정
- **데이터베이스 프로그래밍이란?**
  - DBMS에 데이터를 정의하고 저장된 데이터를 읽어와 데이터를 변경하는 프로그램을 작성하는 과정
  - 일반 프로그래밍과는 데이터베이스 언어인 SQL을 포함한다는 점이 다름



(a) SQL Interface (b) 삽입 프로그래밍

그림 5-1 **데이터베이스 프로그래밍**

01. 데이터베이스 프로그래밍의 개념

- **데이터베이스 프로그래밍 방법**

❶ **SQL 전용 언어를 사용하는 방법**

SQL 자체의 기능을 확장하여 변수, 제어, 입출력 등의 기능을 추가한 새로운 언어를 사용하는 방법.
Oracle은 PL/SQL 언어를 사용하며, SQL Server는 T-SQL이라는 언어를 사용함.

● **일반 프로그래밍 언어에 SQL을 삽입하여 사용하는 방법**

자바, C, C++ 등 일반 프로그래밍 언어에 SQL 삽입하여 사용하는 방법.
일반 프로그래밍 언어로 작성된 응용 프로그램에서 데이터베이스에 저장된 데이터를 관리, 검색함.
삽입된 SQL문은 DBMS의 컴파일러가 처리함.

● **웹 프로그래밍 언어에 SQL을 삽입하여 사용하는 방법**

호스트 언어가 JSP, ASP, PHP 등 웹 스크립트 언어인 경우다.

● **4GL(4th Generation Language)**

데이터베이스 관리 기능과 비주얼 프로그래밍 기능을 갖춘 'GUI 기반 소프트웨어 개발 도구'를 사용하여 프로그래밍하는 방법. Delphi, Power Builder, Visual Basic 등이 있음.

01. 데이터베이스 프로그래밍의 개념

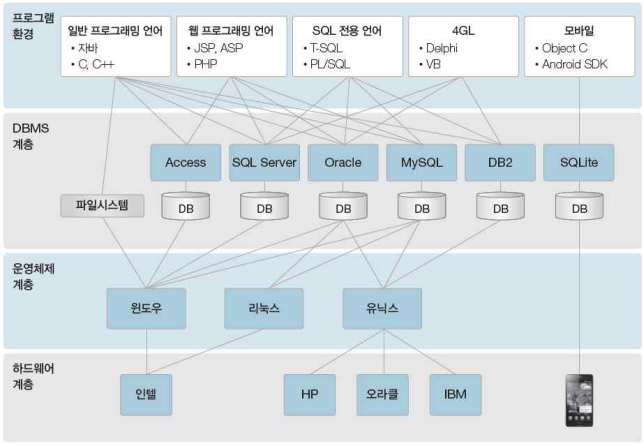


그림 5-2 **DBMS 플랫폼과 데이터베이스 프로그래밍의 유형**

01. 데이터베이스 프로그래밍의 개념

표 5-1 **DBMS의 종류와 특징**

특징	Access	SQL Server	Oracle	MySQL	DB2	SQLite
제조사	마이크로 소프트사	마이크로 소프트사	오라클사	오라클사	IBM사	리처드 힙 (오픈소스)
운영체제 기반	윈도우	윈도우	**윈도우**, 유닉스, 리눅스	윈도우, 유닉스, 리눅스	유닉스	모바일 OS (안드로이드, iOS 등)
용도	개인용 DBMS	윈도우 기반 기업용 DBMS	대용량 데이터 베이스를 위한 응용	소용량 데이터 베이스를 위한 응용	대용량 데이터 베이스를 위한 응용	모바일 전용 데이터베이스

02. PL/SQL

- 프로시저
- 트리거
- 사용자 정의 함수
- PL/SQL 문법 요약

2. PL/SQL

- Procedural Language/Structured Query Language의 줄임말로 데이터베이스 응용 프로그램을 작성하는 데 사용하는 오라클의 SQL 전용 언어.
- SQL 전용 언어로 SQL 문에 변수, 제어, 입출력 등의 프로그래밍 기능을 추가하여 SQL 만으로 처리하기 어려운 문제를 해결함.
- PL/SQL은 SQL Developer에서 바로 작성하고 컴파일한 후 결과를 실행함.

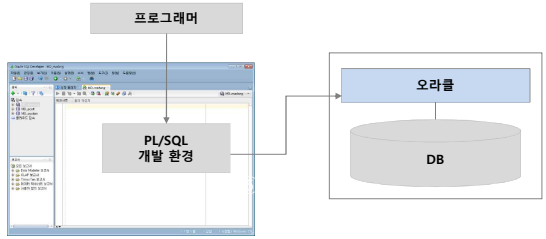


그림 5-3 PL/SQL 개발 환경

2.1 프로시저

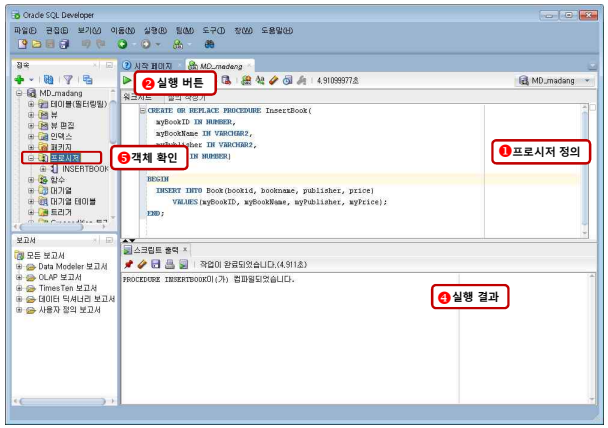


그림 5-4 프로시저를 정의하는 과정

2.1 프로시저

- 프로시저를 정의하려면 CREATE PROCEDURE 문을 사용함.

- 정의 방법
  - P L/SQL은 선언부와 실행부(BEGIN-END)로 구성됨.
    선언부에서는 변수와 매개변수를 선언하고, 실행부에서는 프로그램 로직을 구현함.
  - 매개변수(parameter)는 저장 프로시저가 호출될 때 그 프로시저에 전달되는 값임.
  - 변수(variable)는 저장 프로시저나 트리거 내에서 사용되는 값임.
  - 소스코드에 대한 설명문은 /*와 */ 사이에 기술한다.
    만약 설명문이 한 줄이면 이중 대시(--) 기호 다음에 기술해도 됨.

2.1.1 삽입 작업을 하는 프로시저

- 프로시저로 데이터를 삽입 작업을 하면 좀 더 복잡한 조건의 삽입 작업을 인자 값만 바꾸어 수행할 수도 있고, 저장해두었다가 필요할 때마다 호출하여 사용할 수도 있음.

예제 5-1 Book테이블에 한 개의 투플을 삽입하는 프로시저 (InsertBook)

```
01 CREATE OR REPLACE PROCEDURE InsertBook(
02     myBookID IN NUMBER,
03     myBookName IN VARCHAR2,
04     myPublisher IN VARCHAR2,
05     myPrice IN NUMBER)
06 AS
07 BEGIN
08      INSERT INTO Book(bookid, bookname, publisher, price)
09      VALUES(myBookID, myBookName, myPublisher, myPrice);
10 END;
11
```

```
A /* 프로시저 InsertBook을 테스트하는 부분 */
B EXEC InsertBook(13, '스포츠과학', '마당과학서적', 25000);
C SELECT * FROM Book;
```

2.1.2 제어문을 사용하는 프로시저

- PL/SQL의 제어문은 어떤 조건에서 어떤 코드가 실행되어야 하는지를 제어하기 위한 문법으로, 절차적 언어의 구성요소를 포함함.

표 5-2 PL/SQL의 제어문

구문	의미	문법
BEGIN-END	• PL/SQL 문을 블록화시킴 • 중첩 가능	BEGIN { SQL 문 } END
IF-ELSE	• 조건의 검사 결과에 따라 문장을 선택적으로 수행	IF <조건> SQL 문 [ELSE SQL 문] END IF;
FOR	•counter 값이 범위 내에 있을 경우 FOR 문의 블록을 실행	FOR counter IN <범위> {SQL 문} END LOOP
WHILE	• 조건이 참일 경우 WHILE 문의 블록을 실행	WHILE <조건> { SQL 문 | BREAK | CONTINUE } END LOOP
RETURN	• 프로시저를 종료 • 상태값을 정수로 반환 가능	RETURN [<정수>]

2.1.2 제어문을 사용하는 프로시저

예제 5-2 동일한 도서가 있는지 점검한 후 삽입하는 프로시저(BookInsertOrUpdate)

```
01 CREATE OR REPLACE PROCEDURE BookInsertOrUpdate(
02    myBookID NUMBER,
03    myBookName VARCHAR2,
04    myPublisher VARCHAR2,
05    myPrice INT)
06 AS
07    mycount NUMBER;
08 BEGIN
09   SELECT COUNT(*) INTO mycount FROM Book
10     WHERE bookname LIKE myBookName;
11   IF mycount!=0 THEN
12     UPDATE Book SET price = myPrice
13       WHERE bookname LIKE myBookName;
14   ELSE
15     INSERT INTO Book(bookid, bookname, publisher, price)
16       VALUES(myBookID, myBookName, myPublisher, myPrice);
17   END IF;
18 END;
```

```
A  /* BookInsertOrUpdate 프로시저를 실행하여 테스트하는 부분 */
B  EXEC BookInsertOrUpdate(15, '스포츠 즐거움', '마당과학서적', 25000);
C  SELECT * FROM Book; /* 15번 투플 삽입 결과 확인 */
D  /* BookInsertOrUpdate 프로시저를 실행하여 테스트하는 부분 */
E  EXEC BookInsertOrUpdate(15, '스포츠 즐거움', '마당과학서적', 20000);
F  SELECT * FROM Book; /* 15번 투플 가격 변경 확인 */
```

2.1.2 제어문을 사용하는 프로시저

BOOKID	BOOKNAME	PUBLISHER	PRICE
1	축구의 역사	굿스포츠	7000
2	축구아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000
13	스포츠과학	마당과학서적	25000
15	스포츠 즐거움	마당과학서적	25000

➡

BOOKID	BOOKNAME	PUBLISHER	PRICE
1	축구의 역사	굿스포츠	7000
2	축구아는 여자	나무수	13000
3	축구의 이해	대한미디어	22000
4	골프 바이블	대한미디어	35000
5	피겨 교본	굿스포츠	8000
6	역도 단계별기술	굿스포츠	6000
7	야구의 추억	이상미디어	20000
8	야구를 부탁해	이상미디어	13000
9	올림픽 이야기	삼성당	7500
10	Olympic Champions	Pearson	13000
13	스포츠과학	마당과학서적	25000
15	스포츠 즐거움	마당과학서적	20000

그림 5-6 BookInsertOrUpdate 프로시저를 실행한 후 Book 테이블

2.1.3 결과를 반환하는 프로시저

예제 5-3 Book 테이블에 저장된 도서의 평균가격을 반환하는 프로시저(AveragePrice)

```
01 CREATE OR REPLACE PROCEDURE AveragePrice(
02    AverageVal OUT NUMBER)
03 AS
04 BEGIN
05    SELECT AVG(price) INTO AverageVal FROM Book WHERE price IS NOT
06 NULL;
07 END;
08
```

```
A   /* 프로시저 AveragePrice를 테스트하는 부분 */
B   SET SERVEROUTPUT ON ;
C   DECLARE
D     AverageVal NUMBER;
E   BEGIN
F      AveragePrice(AverageVal);
G      DBMS_OUTPUT.PUT_LINE('책값 평균: '|| AverageVal);
H   END;
```

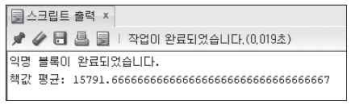


그림 5-7 AveragePrice 프로시저를 실행한 결과

2.1.4 커서를 사용하는 프로시저

- 커서(cursor)는 실행 결과 테이블을 한 번에 한 행씩 처리하기 위하여 테이블의 행을 순서대로 가리키는 데 사용함.

표 5-3 커서와 관련된 키워드

키워드	역할
CURSOR 〈cursor 이름〉 IS 〈커서 정의〉	커서를 생성
OPEN 〈cursor 이름〉	커서의 사용을 시작
FETCH 〈cursor 이름〉 INTO 〈변수〉	행 데이터를 가져옴
CLOSE 〈cursor 이름〉	커서의 사용을 끝냄

2.1.4 커서를 사용하는 프로시저

```
01 CREATE OR REPLACE PROCEDURE Interest
02 AS
03    myInterest NUMERIC;
04    Price NUMERIC;
05    CURSOR InterestCursor IS SELECT saleprice FROM Orders;
06 BEGIN
07   myInterest := 0.0;
08   OPEN InterestCursor;
09   LOOP
10       FETCH InterestCursor INTO Price;
11       EXIT WHEN InterestCursor%NOTFOUND;
12       IF Price >= 30000 THEN
13           myInterest := myInterest + Price * 0.1;
14       ELSE
15           myInterest := myInterest + Price * 0.05;
16       END IF;
17   END LOOP;
18   CLOSE InterestCursor;
19   DBMS_OUTPUT.PUT_LINE(' 전체 이익 금액 = ' || myInterest);
20 END;
21
```

예제 5-4 Orders 테이블의 판매 도서에 대한 이익을 계산하는 프로시저(Interest)

```
A /* Interest 프로시저를 실행하여 판매된 도서에 대한 이익금을 계산 */
B SET SERVEROUTPUT ON;
C EXEC Interest;
```

2.1.4 커서를 사용하는 프로시저

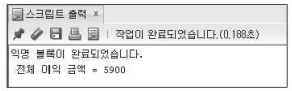


그림 5-8 Interest 프로시저를 실행한 결과

2.2 트리거

- 트리거(trigger)는 데이터의 변경(INSERT, DELETE, UPDATE)문이 실행될 때 자동으로 따라서 실행되는 프로시저를 말함.

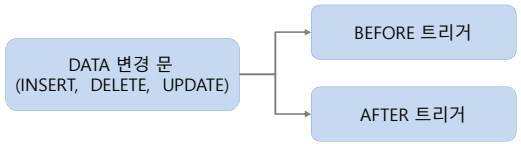


그림 5-9 **데이터 변경과 트리거의 수행**

2.2 트리거

예제 5-5 **새로운 도서를 삽입한 후 자동으로 Book_log 테이블에 삽입한 내용을 기록하는 트리거**

```
A /* 실습을 위한 Book_log 테이블 생성 */
B     CREATE TABLE Book_log(
C        bookid_l NUMBER,
D        bookname_l VARCHAR2(40),
E        publisher_l VARCHAR2(40),
F        price_l NUMBER);
```

```
01 /* 파일명 : AfterInsertBook.sql */
02     CREATE OR REPLACE TRIGGER AfterInsertBook
03     AFTER INSERT ON Book FOR EACH ROW
04     DECLARE
05        average NUMBER;
06     BEGIN
07        INSERT INTO Book_log
08           VALUES(:new.bookid, :new.bookname, :new.publisher, :new.price);
09        DBMS_OUTPUT.PUT_LINE('삽입 투플을 Book_log 테이블에 백업..');
10     END;
```

```
G /* 삽입한 내용을 기록하는 트리거 확인 */
H     INSERT INTO Book VALUES(14, '스포츠 과학 1', '이상미디어', 25000);
I     SELECT * FROM Book WHERE bookid='14';
J     SELECT * FROM Book_log WHERE bookid_l='14'; /* 결과 확인 */
```

2.2 트리거

```
G /* 삽입한 내용을 기록하는 트리거 확인 */
H     INSERT INTO Book VALUES(14, '스포츠 과학 1', '이상미디어', 25000);
I     SELECT * FROM Book WHERE bookid='14';
J     SELECT * FROM Book_log WHERE bookid_l='14'; /* 결과 확인 */
```

```
1개 행 이(가) 삽입되었습니다.
삽입 투플을 Book_log 테이블에 백업..
```

H행 호출 결과

BOOKID	BOOKNAME	PUBLISHER	PRICE
14	스포츠 과학 1	이상미디어	25000

I행 호출 결과

BOOKID_L	BOOKNAME_L	PUBLISHER_L	PRICE_L
14	스포츠 과학 1	이상미디어	25000

J행 호출 결과

그림 5-10 **Book 테이블에 투플을 삽입하여 트리거가 실행된 결과**

2.3 사용자 정의 함수

- 사용자 정의 함수는 수학의 함수와 마찬가지로 입력된 값을 가공하여 결과 값을 되돌려줌.

예제 5-6 **판매된 도서에 대한 이익을 계산하는 함수(fnc_Interest)**

```
01  CREATE OR REPLACE FUNCTION fnc_Interest(
02    price NUMBER) RETURN INT
03  IS
04     myInterest NUMBER;
05  BEGIN
06  /* 가격이 30,000원 이상이면 10%, 30,000원 미만이면 5% */
07    IF Price >= 30000 THEN myInterest := Price * 0.1;
08    ELSE myInterest := Price * 0.05;
09   END IF;
10   RETURN myInterest;
11  END;
12
```

```
A /* Orders 테이블에서 각 주문에 대한 이익을 출력 */
B    SELECT custid, orderid, saleprice, fnc_Interest(saleprice) interest
C    FROM Orders;
```

2.3 사용자 정의 함수

CUSTID	ORDERID	SALEPRICE	INTEREST
1	1	6000	300
1	2	21000	1050
2	3	8000	400
3	4	6000	300
4	5	20000	1000
1	6	12000	600
4	7	13000	650
3	8	12000	600
2	9	7000	350
3	10	13000	650

그림 5-11 Orders 테이블의 건별 이익금 계산

2.3 사용자 정의 함수

표 5-4 프로시저, 트리거, 사용자 정의 함수의 공통점과 차이점

	프로시저	트리거	사용자 정의 함수
공통점	저장 프로시저임, 오라클의 경우 PL/SQL로 작성		
정의 방법	CREATE PROCEDURE 문	CREATE TRIGGER 문	CREATE FUNCTION 문
호출 방법	EXEC 문으로 직접 호출	INSERT, DELETE, UPDATE 문이 실행될 때 자동으로 실행됨	SELECT 문으로 호출
기능의 차이	SQL 문으로 할 수 없는 복잡한 로직을 수행	기본 값 제공, 데이터 제약 준수, SQL 뷰의 수정, 참조무결성 작업 등을 수행	속성 값을 가공하여 반환, SQL 문에 직접 사용

2.4 PL/SQL 문법 요약

표 5-5 PL/SQL의 기본 문법 – http://docs.oracle.com/cd/E11882_01/index.htm

구분	명령어
Data Definition Language (데이터 정의어)	CREATE TABLE CREATE PROCEDURE CREATE FUNCTION CREATE TRIGGER ALTER, DROP
Data Manipulation Language (데이터 조작어)	SELECT INSERT DELETE UPDATE
Data Types(데이터 타입)	NUMBER(n), VARCHAR2(n), DATE
Variables(변수)	DECLARE 문으로 선언 치환(:= 사용)
Operator(연산자)	산술연산자 (+, -, *, /) 비교연산자 (=, <, >, > =, < =, < >) 문자열연산자 (||) 논리연산자 (NOT, AND, OR)
Language Element(주석)	- -, /* */
Built-in Function(내장 함수)	숫자 함수 (ABS, CEIL, FLOOR, POWER 등) 집계 함수 (AVG, COUNT, MAX, MIN, SUM) 날짜 함수 (SYSDATE, NEXT_DAY, TO_CHAR 등) 문자 함수 (CHR, LENGTH, LOWER, SUBSTR 등)
Control of Flow(제어문)	BEGIN-END IF-THEN-ELSE FOR LOOP-END LOOP WHILE LOOP-END LOOP, EXIT
Data Control Language (데이터 제어어)	GRANT REVOKE

03 데이터베이스 연동 자바 프로그래밍

표 5-6 데이터베이스 연동 자바 프로그래밍 실습 환경

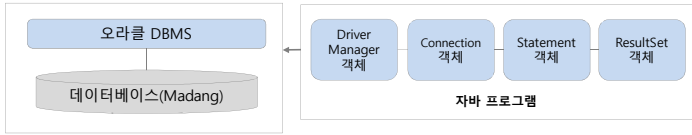
항목	프로그램
데이터베이스 프로그램	오라클 11g r2
자바 컴파일러	JDK 버전 7
데이터베이스와 자바를 연결하는 드라이버	JDBC 드라이버(파일이름 ojdbc6.jar)

3.1 소스코드 설명

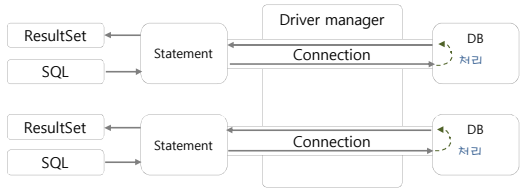
표 5-7 데이터베이스 접속 자바 클래스(java.sql)

클래스 구분	클래스 혹은 인터페이스	주요 메소드 이름	메소드 설명
java.lang	Class	Class forName(<클래스이름>)	<클래스이름>의 JDBC 드라이버를 로딩
java.sql	DriverManager	Connection getConnection (url, user, password)	데이터베이스 Connection 객체를 생성
	Connection	Statement createStatement()	SQL 문을 실행하는 Statement 객체를 생성
		void close()	Connection 객체 연결을 종료
	Statement	ResultSet executeQuery (String sql)	SQL 문을 실행해서 ResultSet 객체를 생성
		ResultSet executeUpdate (String sql)	INSERT/DELETE/UPDATE 문을 실행해서 ResultSet 객체를 생성
	ResultSet	boolean first()	결과 테이블에서 커서가 처음 투플을 가리킴
		boolean next()	결과 테이블에서 커서가 다음 투플을 가리킴
		int getInt(<int>)	<int>가 가리키는 열 값을 정수로 반환
		String getString(<int>)	<int>가 가리키는 열 값을 문자열로 반환

3.1 소스코드 설명



(a) 자바의 데이터베이스 연동 객체



(b) 객체 간의 호출 순서

그림 5-12 데이터베이스 연결 자바 객체들의 호출 관계

3.2 프로그램 실습

표 5-8 자바 프로그램 실습 단계

단계		세부 단계	프로그램
[1단계] DBMS 설치 및 환경설정		① 오라클 11g r2 설치 ② 오라클 접속을 위한 사용자(madang) 생성	오라클 11g
[2단계] 데이터베이스 준비		① 마당서점 데이터베이스 준비 (demo_madang.sql)	
[3단계] 자바 실행	(A) 명령 프롬프트 이용	① 자바 컴파일러 설치 ② JDBC 드라이버 설치 ③ 자바 프로그램 준비(booklist.java) ④ 컴파일 및 실행	JDK JDBC
	(B) 이클립스 이용	① 자바와 이클립스 개발도구 설치 ② JDBC 드라이버 설치 ③ 자바 프로그램 준비(booklist.java) ④ 컴파일 및 실행	JDK Eclipse JDBC

3.2 프로그램 실습

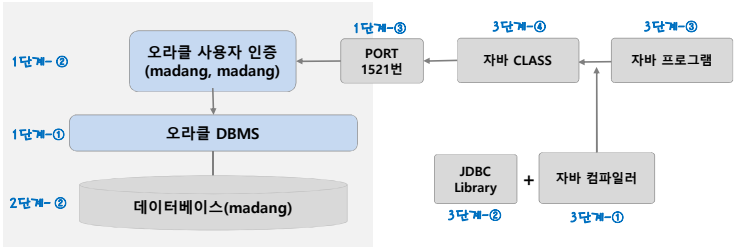


그림 5-14 데이터베이스 연동 자바 프로그램의 실행 흐름도

4.1 소스코드 설명

- JSP 프로그램은 HTML 태그에 JSP 스크립트를 끼워 넣어 작성하는데, JSP 스크립트 부분은 <% ... %>에 넣어서 실행시킴.

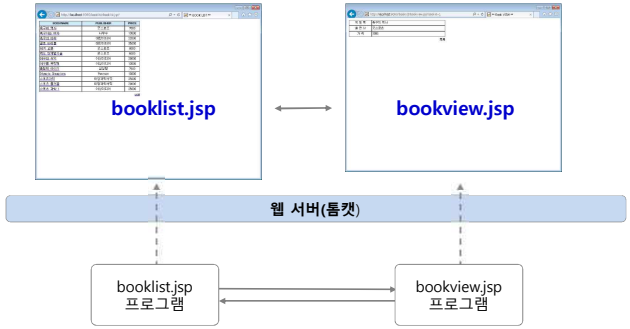


그림 5-24 booklist.jsp와 bookview.jsp의 호출 관계와 웹에서 실행된 화면

4.2 프로그램 실습

④ 실행

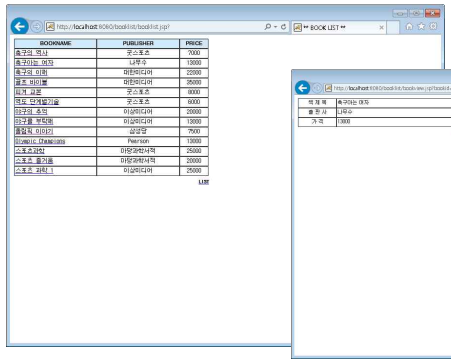


그림 5-23 booklist.jsp 실행 화면

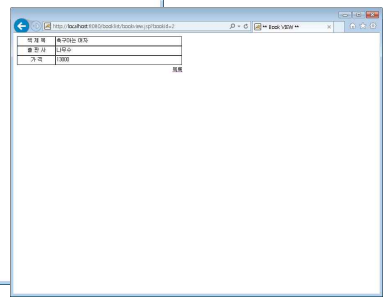


그림 5-24 bookview.jsp 실행 화면