Oracle11g chapter 18 PL/SQL

${} 자주 바뀌지 않거나 사용자의 입력을 받는 경우 (ex 테이블 명 , 컬럼 명) 에 사용해 주면 좋다.

#{} 은 사용자의 입력을 받거나 데이터가 많은 경우에 주로 사용한다.

PL/SQL(ORACLE’S PROCEDURAL LANGUAGE EXTENSION TO SQL)

SQL문장에서 변수정의, 조건처리, 반복처리등을 지원

1.SET SERVEROUTPUT ON // 환경변수의 DEFAULT 값이 OFF이기 때문에 ON으로 변경하여야한다

->사용할려는 DBMS\_OUTPUT.PUT\_LINE프로시저를 사용하기위해

2.DECLARE// 변수정의

변수나 상수선언

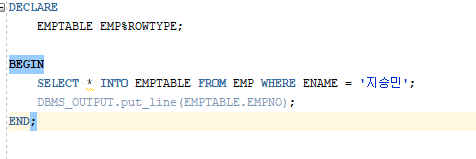
-변수선언방법

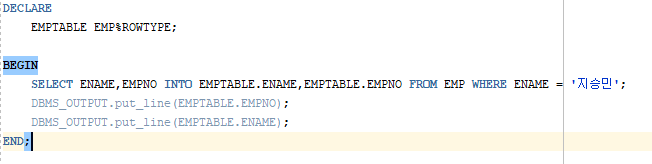
변수이름 [CONSTANT] DATATYPE[NOT NULL]

CONSTATNT = 변수의값을 변경할수없도록 제약(상수)

DATATYPE으로 NUMBER,VARCHAR2 …등이오며 EMP%ROWTYPE(EMP테이블자체를저장할려 ROWTYPE선언), EMP.EMPNO%TYPE(EMP테이블을 작성할 때 EMPNO의 자료형을 참조하여 사용할려는 변수로 선언시켜준다)

이때 SELECT문의 대상칼럼을 지정해주면 INTO안에 ROWTYPE의 변수에도 각각의 칼럼을 지정해주어야하고 \*로통일시 변수이름만 작성하여도된다





또한 ROWTYPE은 하나의 행만을 가져와 저장할수있어 WHERE조건절의 해당하는 행을 지정해줘야한다

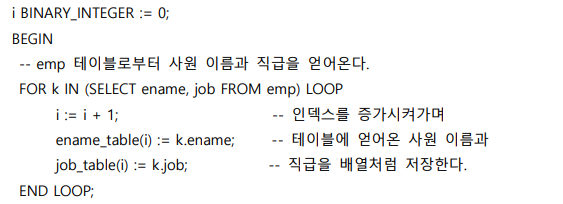
테이블타입을 정의

TYPE 테이블타입이름 IS TABLE OF 해당테이블칼럼의 타입

INDEX BY BINARY\_INTEGER;

-> 변수 테이블타입; //정의한 테이블타입으로 변수를 선언할수있음

또한 I BINARY\_INTEGER :0;을 선언하여 아래와같이 다음과 같은 FOR문을 실행시킬수있다



RECORD TYPE : 프로그램 언어의 구조체와 유사

TYPE 레코드타입이름 IS RECORD

(레코드내의 필드이름 해당칼럼%TYPE,

레코드내의 필드이름 해당칼럼%TYPE,

레코드내의 필드이름 해당칼럼%TYPE

…);

변수이름 레코드타입이름; // 해당 레코드타입이름으로 변수를 선언할수있음 //ROWTYPE 과 유사

또한

3.BEGIN //조건처리, 반복처리 끝날때는 구문의 끝을 알리는 END;

제어문( IF ~ THEN ~ END IF)

-1가지의 조건에대한 처리

IF 조건 THEN

조건을 만족하면 처리;

ENDIF

-2가지 조건에대한처리

IF 조건 THEN

조건을 만족하면 처리;

ELSE

조건을 만족하지 않으면 처리;

END IF;

-여러가지의 조건에 대한처리

IF 조건 THEN

조건을 만족하면 처리;

ELSIF 조건 THEN

조건을 만족하면 처리;

ELSIF 조건 THEN

조건을 만족하면 처리;

ELSE

위의조건들이 모두 아닐시 처리;  
END IF;

반복문

-조건없이 반복 작업을 제공하는 BASIC LOOP문(EXIT를통해 반복문 탈출

LOOP

실행

IF 조건 THEN

EXIT;

END IF;

-COUNT를 기본으로 작업의 반복 제어를 제공하는 FOR LOOP문

FOR 변수 IN [REVERSE] (SELECT 가져올칼럼명 FROM 가져올테이블명 or n..k) LOOP

실행문

END LOOP;

이때 변수는 암시적으로 선언된 변수로 테이블을 IN()안에 테이블을 가져온다면

변수 EMP%ROWTYPE;

SELECT \* INTO 변수 FROM EMP;를 해준것과 같은 의미  
따라서 FOR문의 IN안에 테이블을 가져왔다면 실행문에서 변수.해당칼럼하여 값을 가져올수있으며 변수의 모든 값을 가져올때까지 FOR문이 실행된다

-WHILE LOOP문 제어조건이 TURE인 동안만 반복 제어조건이 FALSE이게되면 반복문장을 탈출

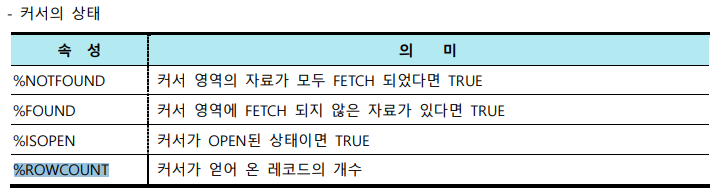
WHILE 제어조건 LOOP

STATEMENT1;

STATEMENT2;

END LOOP;

-CURSOR : 처리결과가 여러 개의 행으로 구해지는 SELECT문을 처리



DECLARE

EMP01R EMP%ROWTYPE;

CURSOR CR1 IS SELECT \* FROM EMP;

BEGIN

OPEN CR1;

LOOP

FETCH CR1 INTO EMP01R.EMPNO,EMP01R.ENAME,EMP01R.HIREDATE ;

EXIT WHEN CR1%NOTFOUND;

DBMS\_OUTPUT.PUT\_LINE(EMP01R.EMPNO||EMP01R.ENAME||EMP01R.HIREDATE||' '||CR1%ROWCOUNT);

END LOOP;

DBMS\_OUTPUT.PUT\_LINE(EMP01R.EMPNO||EMP01R.ENAME||EMP01R.HIREDATE||' '||CR1%ROWCOUNT);

CLOSE CR1;

END;

이때 WHILE LOOP로 실행하게되면 FETCH자체가 실행되기전에 조건을 검사하므로 LOOP자체가 돌아가지않는다

저장 프로시저 : 자주사용되는 쿼리문을 모듈화시켜 필요할때마다 호출하여 사용

CREATE [OR REPLACE] PROCEDURE 프로시저이름( 매개변수이름 [MODE] DATA TYPE…)

IS

변수 선언부

BEGIN

-

END;

생성한 저장프로시저는 EXECUTE 명령어로 실행시킨다

[OR REPLACE] : 먼저생성된 프로시저가있다면 변경, 없다면 재생성하는 옵션

ORACLE11 PAGE 132 [MODE] : IN/OUT/IN OUT으로 세종류가 있으며

IN : 프로시저로 값을 전달하는 MODE

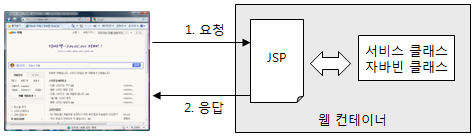
OUT : 프로시저 실행후 해당 매개변수로 OUT해주는 MODE

IN OUT : 매개변수를 통해 결과값을 입력 받고 프로시저 수행후 변형돤 DATA를 받는형태

(주로 단위변환사이에서 사용됨)

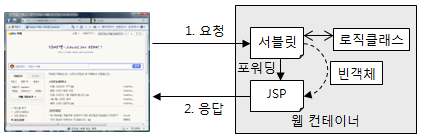
Jsp593~769 P106 MVC패턴

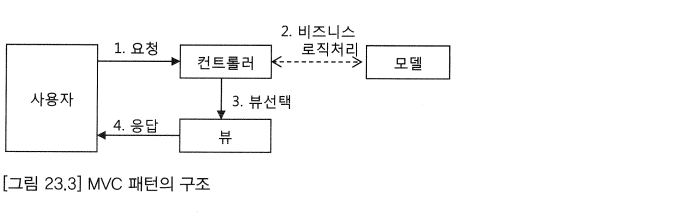
-모델1 구조



웹브라우저의 요청을 jsp가 받아 작업을 처리하고 그결과를 출력해준다

-모델2 구조





웹브라우저의 요청을 서블릿이받아 서블릿이 요청을 알맞게 처리한후 그결과를 jsp페이지로 포워딩한다

#MVC(Model-View-Controller)

Model : 비즈니스 영역의 상태정보를 처리한다

View : 비즈니스 영역에 대한 프레젠테이션 뷰(사용자가 보게될 결과 화면)를 담당한다

Controller : 사용자의 입력 및 흐름 제어를 담당한다

MVC 패턴의 핵심

-비즈니스 로직을 처리하는 모델과 결과화면을 보여주는 뷰가 분리되어있다

-어플리케이션의 흐름 제어나 사용자의 처리요청은 컨트롤러에 집중된다

-따라서 모델의 내부로직을 변경하더라도 뷰는 영향을 받지 않는다 -> 뷰와 모델이 결합되어있지 않기 때문에 뷰를 어렵지 않게 변경할수있다

- 유연하고 확장하기 쉽다

- 디자이너와 개발자의 협업이 용이하다.

- 유지보수 비용을 절감할 수 있다.

# MVC의 단점

- 기본기능 설계를 위해 클래스들이 많이 필요하기 때문에 복잡할 수 있다.

- 설계시간이 오래 걸리고 숙련된 개발자가 필요하다.

- Model과 View의 완벽한 분리가 어렵다.

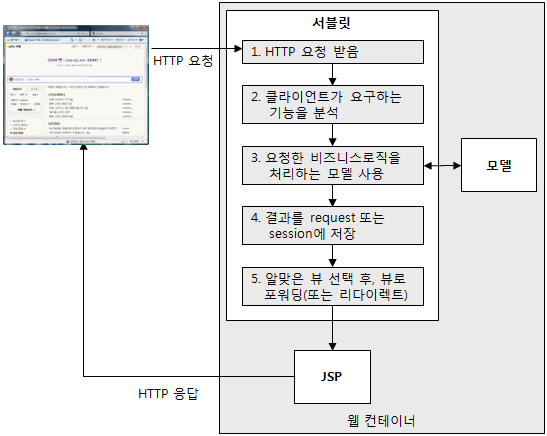
컨트롤러 = 서블릿

모델 = 비즈니스 로직처리 클래스.자바빈

뷰 = jsp

사용자 = 웹브라우저, 휴대폰과 같은 다양한 기기

#MVC컨트롤러: 서블릿



과정1: 웹브러우저가 전송한 HTTP요청을 받는다. 서블릿의 doGet()메서드나 doPost()메서드가 호출된다

과정2: 웹브라우저가 어떤 기능을 요청했는지 분석한다 -> 게시판목록을 요청하였는지,글쓰기를 요청하였는지 알아낸다

과정3: 모델을 사용하여 요청한 기능을 수행한다

과정4: 모델로부터 전달받은 결과물을 알맞게 가공한후 request나 session의 setAttribute()메서드를 사용하여 결과값을 속성에 저장한다. 이렇게 저장된 결과값은 뷰인 JSP에서 사용된다

과정5: 웹브라우저에 보여줄 JSP를 선택한후, 해당 JSP로 포워딩한다. 경우에 따라서 리다이렉트를 하기도한다

#MVC의 모델

