[11/28] Oracle SQL, MVC pattern

1) java 에 관하여[1]

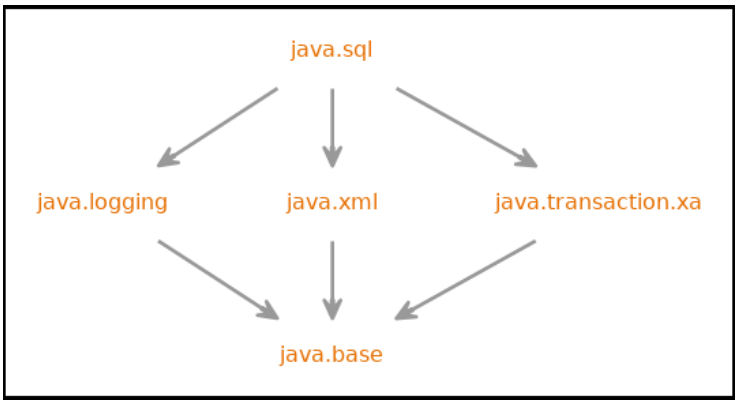
- version
  - classic version 은 개발자 5%미만이 여전히 채택중, 사고 및 예외변수 미발생
  - java8(jdk1.8) : php jsp 단독서버 운영시
  - java8, java11 : spring, springboot 는 jsp 파일 사용
  - java17 : springboot 3.0 은 상위 버전 필요

- module
  - 모든 라이브러리를 한번에 호출하도록 * 설정

```
<%@ page import="java.sql.*" %>
```

    - 정확한 import 가 구동시간을 줄여주나 현 단계에서는 간략히 전체(*) 호출

  - java > API Documentation > java.sql




*module java.sql*

  - java의 구동 원리를 알아야 하나 시간상 생략…

- platform[2]
  - Java SE (Standard Edition)
    - 표준 에디션 : 데스크톱, 자바, 임베디드를 위한 플랫폼
  - Java EE (Enterprise Edition)
    - 과거 보편적인 IDE 툴인 eclipse등 자바를 이용한 서버측 개발을 위한 플랫폼
    - Java EE는 Java SE에서 API(lib 디렉토리에 포함되어 있는 JAR 파일들)가 추가된 것
  - Java ME (Micro Edition) : 임베디드를 위한 자바 플랫폼
  - JavaFX : 경량 사용자 인터페이스 API를 사용하여 리치 인터넷 app을 만들 때 사용

[1] Java Platform, Standard Edition Documentation
https://dev.java/learn/

[2] [JAVA] Java SE 와 Java EE 차이점
[JAVA] - Java SE, Java EE 차이 간단하고 쉽게 이해하기(자바 플렛폼의 종류)

2) golf - teacher.jsp

- JDBC 와의 연결 구문

```
try{
    Class.forName("oracle.jdbc.OracleDriver");
    Connection con = DriverManager.getConnection
                    ("jdbc:oracle:thin:@//localhost:1521/xe","system","1234");
```

body 에 포함

- rs.getString() : 해당 순서의 열에있는 데이터를 String형으로 받아옴

```
<td><%=rs.getString("teacher_code") %></td>
```

3) sql 구문 연습(정보처리기사문제중)

- where C between A and B : A와 B 사이의 C 를 호출

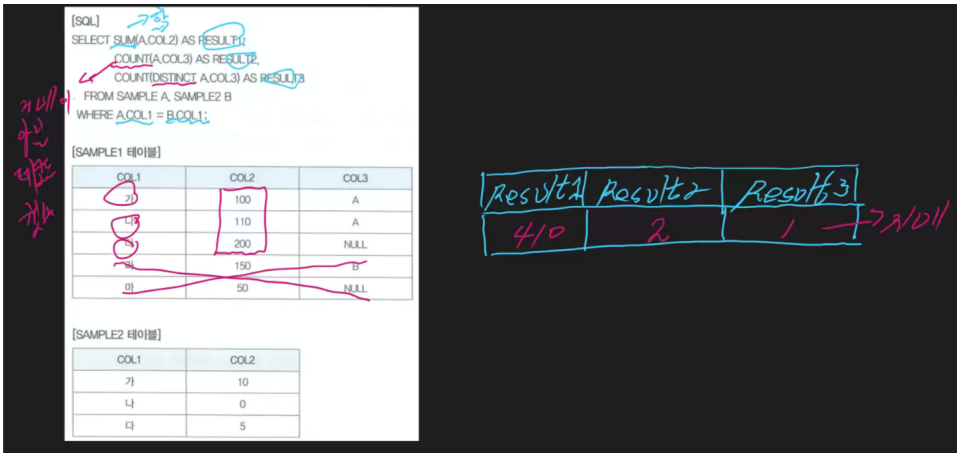
```
SELECT ENAME, SAL
  FROM EMP
ORDER BY SAL;
```

| | ENAME | SAL |
|---|---|---|
| 1 | SMITH | 800 |
| 2 | JAMES | 950 |
| 3 | WOKJK | 1250 |
| 4 | WORRY | 1250 |
| 5 | WOWO | 1250 |
| 6 | MARTIN | 1250 |
| 7 | WARD | 1250 |
| 8 | MILLER | 1300 |
| 9 | TURNER | 1500 |
| 10 | ALLEN | 1600 |
| 11 | CLARK | 2450 |
| 12 | BLAKE | 2850 |
| 13 | JONES | 2975 |
| 14 | FORD | 3000 |
| 15 | KING | 5000 |

```
SELECT ENAME, SAL
  FROM EMP
WHERE SAL BETWEEN 800 AND 1500
ORDER BY SAL;
```

| | ENAME | SAL |
|---|---|---|
| 1 | SMITH | 800 |
| 2 | JAMES | 950 |
| 3 | WOKJK | 1250 |
| 4 | MARTIN | 1250 |
| 5 | WOWO | 1250 |
| 6 | WORRY | 1250 |
| 7 | WARD | 1250 |
| 8 | MILLER | 1300 |
| 9 | TURNER | 1500 |

- 예제2




  - sum( ) : ( ) 안의 합, Result : 해당 테이블 아래에 출력
  - count : null 이 아닌 갯수 / count(distinct : null 의 갯수

- 예제3 : 다음 두 테이블을 COL1을 JOIN 컬럼으로 하여 LEFT OUTER JOIN, RIGHT JOIN, Full OUTER JOIN 했을 때 각각 출력되는 데이터 건수로 가장 적절한 것은 무엇인가?(단, SAMPLE1 테이블이 LEFT TABLE, SAMPLE2 테이블이 RIGHT TABLE 이라고 가정)

[SAMPLE1 테이블]

| COL1 | COL2 | COL3 |
|---|---|---|
| 1 | A | D |
| 3 | B | E |
| 5 | C | F |

[SAMPLE2 테이블]

| COL1 | COL4 |
|---|---|
| 2 | G |
| 3 | H |
| 3 | I |

  - right outer join :
    - ```
      select a.col1, a.col2, a.col3, b.col1, b.col4  // (3) 해당 순서로 출력
      from sample2 b
      right outer join sample1 a // (1) right 므로 sample1은 무조건 출력
      on a.col1=b.col1; // (2) 붙일때 이 조건에 해당되는 애들만 붙음, 나머진 null
      ```

| | COL1 | COL2 | COL3 | COL1_1 | COL4 |
|---|---|---|---|---|---|
| 1 | 3 | B | E | 3 | H |
| 2 | 3 | B | E | 3 | I |
| 3 | 1 | A | D | (null) | (null) |
| 4 | 5 | C | F | (null) | (null) |

  - Left outer join :
    - ```
      select a.col1, a.col2, a.col3, b.col1, b.col4 // (3) 해당 순서로 출력
      from sample2 b // (1) left 므로 sample2 은 무조건 출력
      left outer join sample1 a
      on a.col1=b.col1;  // (2) 붙일때 이 조건에 해당되는 애들이 붙음,  나머진 null
      ```

| | COL1 | COL2 | COL3 | COL1_1 | COL4 |
|---|---|---|---|---|---|
| 1 | 3 | B | E | 3 | I |
| 2 | 3 | B | E | 3 | H |
| 3 | (null) | (null) | (null) | 2 | G |

  - full outer join : 기준 테이블은 그대로, 기준 테이블에서 없는 내용은 null
    - ```
      select a.col1, a.col2, a.col3, b.col1, b.col4
      from sample2 b
      full outer join sample1 a
      on a.col1=b.col1;
      ```

| | COL1 | COL2 | COL3 | COL1_1 | COL4 |
|---|---|---|---|---|---|
| 1 | 1 | A | D | (null) | (null) |
| 2 | 3 | B | E | 3 | I |
| 3 | 3 | B | E | 3 | H |
| 4 | 5 | C | F | (null) | (null) |
| 5 | (null) | (null) | (null) | 2 | G |
      ```

- 예제4 :

```
[SQL]
SELECT *
  FROM SAMPLE1 A RIGHT OUTER JOIN SAMPLE2 B
    ON (A.COL1 = B.COL1 AND B.COL2 IS NOT NULL);
```

[SAMPLE1 테이블]

| COL1 | COL2 |
|---|---|
| 2 | G |
| 3 | H |
| 3 | I |

[SAMPLE2 테이블]

| COL1 | COL2 |
|---|---|
| 1 | A |
| 2 | B |
| 3 | NULL |
| 4 | D |
| 5 | E |

| | COL1 | COL2 | COL1_1 | COL2_1 |
|---|---|---|---|---|
| 1 | (null) | (null) | 1 | A |
| 2 | 2 | G | 2 | B |
| 3 | (null) | (null) | 3 | (null) |
| 4 | (null) | (null) | 4 | D |
| 5 | (null) | (null) | 5 | E |

  - right outer join sample2 가 가장 우선, sample2 는 전체 호출
  - on(A.COL1 = B.COL1 AND B.COL2 is not null) 에 해당하는 조건이 붙음
    - sample1(2 G) 와 sample2 (2 B) 가 해당되나,
    - 현재의 기준인 sample2 에게 sample1의 (2 G) 만 붙게 됨
  - select 조건이 언급되지 않으면 on 의 조건부터 가장 먼저 출력
  - 주의 사항 : null 입력시 string 으로 입력하지 않도록 주의

- 예제5 :

```
[SQL]
SELECT *
  FROM SAMPLE1 A RIGHT OUTER JOIN SAMPLE2 B
    ON A.COL1 = B.COL1
 WHERE B.COL2 IS NOT NULL;
```

[SAMPLE1 테이블]

| COL1 | COL2 |
|---|---|
| 2 | G |
| 3 | H |
| 3 | I |

[SAMPLE2 테이블]

| COL1 | COL2 |
|---|---|
| 1 | A |
| 2 | B |
| 3 | NULL |
| 4 | D |
| 5 | E |

| | COL1 | COL2 | COL1_1 | COL2_1 |
|---|---|---|---|---|
| 1 | 2 | G | 2 | B |
| 2 | (null) | (null) | 1 | A |
| 3 | (null) | (null) | 4 | D |
| 4 | (null) | (null) | 5 | E |

  - 위의 결과값에서 where 조건에 해당하는 3, null 라인만 사라짐
  - on 의 내용이 가장 먼저 출력

- 예제6 :
  - 해당 구문은 inner join or outer join 이 아니므로 null 이 나오지 않음
  - result : col1(2,4) * col2(55, 30, 20)
  - result : 2 55 / 2 30 / 2 20 / 4 55 / 4 30 / 4 20

다음 SQL의 결과로 가장 적절한 것은 무엇인가?

[SQL]

```
SELECT A.COL1, B.COL3
  FROM SAMPLE1 A, SAMPLE2 B
 WHERE B.COL3 % 5 = 0;
```

[SAMPLE1 테이블]

| COL1 | COL2 |
|---|---|
| 2 | G |
| 4 | I |

[SAMPLE2 테이블]

| COL1 | COL3 |
|---|---|
| 1 | 55 |
| 2 | 12 |
| 3 | 30 |
| 4 | 33 |
| 4 | 20 |

- 예제7 :

다음 SQL 중 에러가 발생하지 않는 SQL은 무엇인가? (Oracle이라고 가정)

① 
```
SELECT A.COL1, B.COL2
  FROM SAMPLE1 A, SAMPLE2 B
 WHERE B.COL2 IS NULL;
```

② 
```
SELECT A.COL1, B.COL2
  FROM SAMPLE1 A NATURAL JOIN SAMPLE2 B
    ON A.COL1 = B.COL1;
```

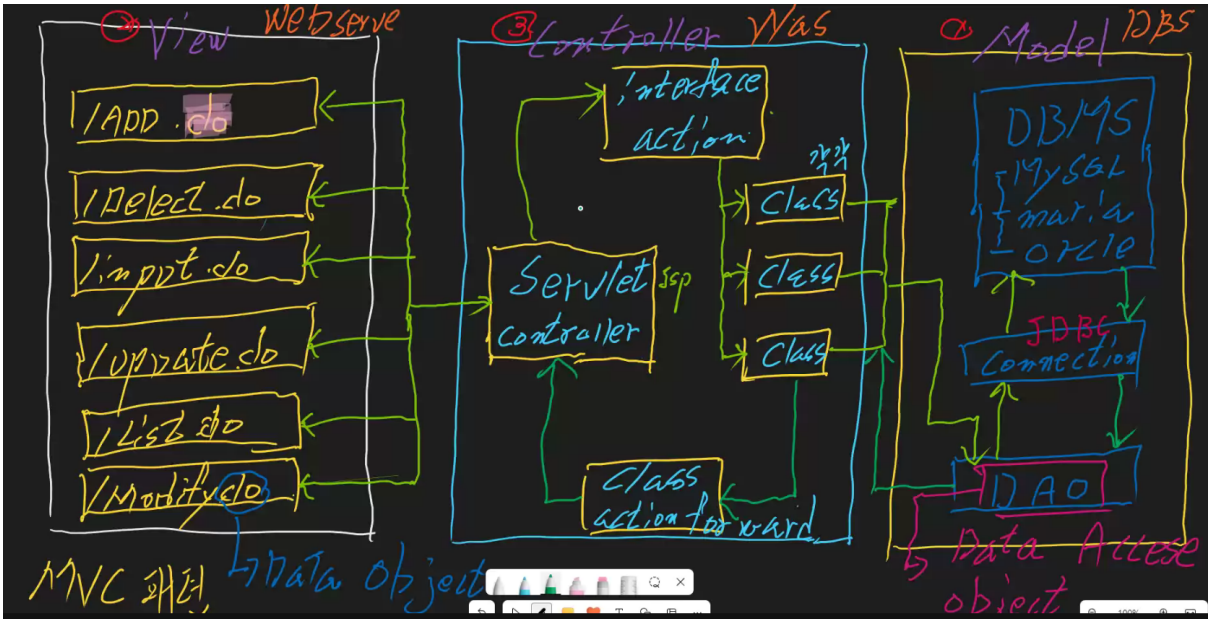
③ 
```
SELECT A.COL1, B.COL2
  FROM SAMPLE1 A, SAMPLE2 B
 WHERE A.COL1(+) = B.COL1(+);
```

④ 
```
SELECT B.COL1, B.COL2
  FROM SAMPLE1 A JOIN SAMPLE2 B
 USING (COL1, COL2);
```

  - natural join 은 where 과 함께 사용 (on 사용하면 안됨)
  - 좌측의 + 는 right outer join, 우측의 +는 left outer join 이며, 두개를 동시에 사용불가
  - using 특정 테이블을 지정, 그러나 join 에서는 using 사용 불가

4) MVC 패턴

- model : dbserver / view : webserver / controller : was





유튜브 시청 필요

- 자료의 호출 순서
  - Client 의 Do 요청을 Controller(Servlet)의 요청에 따른 Interface를 선택
  - 해당 Interface는 실행할 Class를 선택
  - DAO 는 Controller 의 Class 요청(SQL 요청)을 받고 실행하여 순회
  - Do 는 결국 DAO > JDBC > DBMS > JDBC > DAO 로 순회하여 도착
  - DAO 의 결과는 다시 class 를 거쳐 결과 view에 경로 선택(어떤 do를 실행할지)
  - DOM 과 BOM을 선택이 됨

- DO(Data Object)
- DTO(Data Transfer Object) : 계층간 데이터 교환을 위해 사용하는 객체
  - was에서 controller 와 controller 간, controller 와 class간의 통신 송수신의 규격
  - 각각의 사각틀들은 하나의 class(하나의 작업 단위) 로 인식?!
- DAO(Data Access Object)