

Mybatis 기본 사용법

Mybatis는 데이터베이스와 자바프로그램을 연결시켜주는 프레임워크이다.

다시 말해서 데이터베이스의 쿼리 결과를 자바로 가져오고 싶거나, 자바에서 쿼리를 실행하기 위한 툴이다.

그렇기 때문에 **Mabatis를 잘 활용하기 위해서는 1.자바에서 쿼리를 실행하는 방법과 2.쿼리 결과를 자바로 가져오는 방법을 잘 알아야한다.** 그래서 해당 ppt에서는 위에서 언급한 내용을 정리해보겠다.

테이블과 VO클래스 , Mapper.xml, 테이블과의 관계 (resultMap 정의)

VO클래스 하나는 DB 테이블 하나와 매칭되는 것이 일반적이다.

예)

DB에 존재하는 테이블 : BOARD, MEMBER

만들어야 할 VO클래스 : BoardVO, MemberVO

또한, VO클래스는 변수로 테이블의 컬럼에 해당하는 모든 정보를 가지고 있어야 한다.

예) BOARD 테이블에 TITLE, WRITER, CONTENT 라는 컬럼이 존재 한다면 BoardVO클래스는 변수를 아래와 같이 지녀야 한다.

```
Class BoardVO{  
    private String title;  
    private String writer;  
    private String content;  
}
```

그리고 모든 변수 각각에 대한 getter와 setter가 필요하다.

다시 말하지만 자바에서 변수는 소문자로 시작한다. 그리고 합성어라면 boardTitle처럼 카멜표기법을 사용한다.

예)컬럼명 : MEMBER_ID -> 변수 : memberId

하지만 이렇게 테이블에 매칭되는 클래스만 정의한다고 해서 DB의 테이블과 VO객체가 매칭되는 것은 아니다. 테이블과 클래스를 연결시켜주는 매개체가 필요한데 이것을 mapper.xml 파일에서 정의하는 resultMap이 해준다. (조회 쿼리 시 필요!)

```
<resultMap type="vo.BoardVO" id="board">  
    <result column="BOARD_NUM" property="boardNum"/>  
    <result column="BOARD_WRITER" property="boardWriter"/>  
    <result column="BOARD_PASSWORD" property="boardPassword"/>  
    <result column="BOARD_SUBJECT" property="boardSubject"/>  
    <result column="BOARD_CONTENT" property="boardContent"/>  
    <result column="BOARD_FILE" property="boardFile"/>  
    <result column="BOARD_RE_REF" property="boardReRef"/>  
    <result column="BOARD_RE_NUM" property="boardReNum"/>  
    <result column="BOARD_READ_COUNT" property="boardReadCount"/>  
    <result column="BOARD_DATE" property="boardDate"/>  
</resultMap>
```

<result> 태그에서 column 속성의 값은 테이블의 컬럼명과 일치해야 하며, property 속성의 값은 vo클래스의 변수명과 일치해야 된다.

그리고 <result> 태그에 이 둘을 같이 써줌으로서 매칭이 이루어지는 것이다.

자바에서 쿼리를 실행하는 방법(1)

```
<select id="selectBoard" resultMap="board">
    SELECT BOARD_TITLE
      , BOARD_WRITER
      , BOARD_CONTENT
      , BOARD_DATE
      , READ_CNT
      , BOARD_NUM
      , BOARD_FILE
    FROM BOARD
    WHERE BOARD_NUM = #{boardNum}
</select>
```

```
@Override
public BoardVO selectBoard(BoardVO boardVO) {
    BoardVO result = sqlSession.selectOne("selectBoard", boardVO);
    return result;
}
```

```
BoardVO vo = service.selectBoard(boardVO);
```

위의 이미지는 왼쪽에서부터 차례대로 쿼리를 작성한 mapper.xml, 작성한 쿼리를 실행하는 메소드를 작성한 serviceImpl.java, 작성한 메소드를 호출하는 Controller.java의 일부분을 캡처한 것이다. 이전 ppt에서도 언급하였듯 웹 개발에서 데이터베이스 작업이 필요할 경우 코드 작성 순서는 위의 이미지 순서와 동일하다(mapper→service→controller 순서로 코드 작성). 위의 이미지처럼 코드를 구현하는 이유와 방법 또한 이전 ppt에 설명이 되어 있으니 그에 대한 설명은 생략하기로 한다.

이번에는 위의 일련의 과정으로 쿼리를 실행시키고 실행결과를 자바로 가져오는 과정 중 쿼리를 실행시키는 방법에 대해 좀 더 정확히 알아보도록 하겠다.

쿼리를 실행시키기 위해서 우선적으로 필요한 능력이 쿼리 작성 능력이다. 개발자가 구현하고 싶은 기능에 대해 쿼리를 작성하지 못하면 제대로 쿼리를 실행시킬 수 없고, 프로그램을 완성하지 못한다.

실행시킬 쿼리를 작성할 수 있다면, 쿼리 작성 후 쿼리 작동을 위해 자바에서 parameter로 받아와야 하는 데이터를 한 눈에 파악할 수 있다. 위의 이미지를 예로 들자면 작성한 쿼리에 `#{boardNum}`이 보인다. 이처럼 쿼리문에서 `#{}`로 작성된 부분이 자바에서 우리가 parameter로 받아와야 하는 데이터이다.

정리하자면 위 이미지의 'selectBoard'라는 id를 가진 쿼리를 실행시키기 위해서 우리는 boardNum값을 쿼리 실행 시 parameter로 넘겨줘야 한다는 뜻이다.

마지막으로 해당 쿼리를 실행시키기 위해 Mybatis에서 제공하는 메소드 중 어떤 메소드를 사용해야하는지만 판단하면 된다.

자바에서 쿼리를 실행하는 방법(2)

이런 일련의 모든 과정(자바에서 쿼리를 실행시키고 쿼리 결과를 자바에서 받아오는 행위)에서 **가장 우선시 되어야 하는 것이 쿼리이다.**

원하는 정보를 얻기 위해 어떤 쿼리를 사용해야 할지 판단할 수 있고, SQL 문법대로 쿼리문을 작성 할 수만 있다면 50%이상은 할 수 있다.

웹 개발에서 첫번째로 생각할 것은 화면이다. 화면을 어떻게 그리고 어떤 데이터를 보여줄지 구상해야 한다. 이는 평소 웹 서핑 경험을 토대로 사용자가 접근하기 쉽고, 화면을 보면 누구나가 이해할 수 있게 구성하면 된다. 그러니 개인의 경험을 토대로 머리 속으로 화면을 구상하고 어떤 정보를 화면에 뿌려줄지 생각하면 되는 것이기에 현재는 별도로 학습 할 이유는 없다.

사용자에서 보여줄 정보와 화면의 UI를 생각했다면 두번째로 생각해야 하는 것이 쿼리이다. 쿼리를 통해 본인이 화면에 보여주고자 하는 데이터를 화면으로 가져올 수 있다. 쿼리를 작성해야 원하는 데이터를 저장하고 수정하고 삭제할 수 있다.

다시 말하지만 화면 구상은 정답이 없기 때문에 누구나가 본인의 생각대로 화면을 표현하면 되는 것이고, 가장 중요한 것은 보여주고자 하는 데이터를 DB에서 어떻게 가져와야 하는지 아는 것이다. 그러니 어떤걸 만들어야 할지, **코드에서 무엇부터 시작해야 할지 모르겠다면 본인이 보여주고자하는 화면을 생각한 후 무조건 쿼리부터 생각해라.** 쿼리만 작성할 수 있으면 나머지는 어렵지 않다.

쿼리를 작성할 수 있다면 이번에 생각해야 하는 것이 해당 쿼리를 Mybatis의 어떤 메소드로 호출해줄지 판단하는 것이다.

자바에서 쿼리를 실행하는 방법(3)

Mybatis의 메소드 선택방법

다시 말하지만 가장 우선적으로 생각할 것은 쿼리이다. 일단 무조건 실행할 쿼리문을 작성하자. 쿼리를 작성했다면 Mybatis의 메소드 중 어떤 메소드를 실행할지 판단한다. Mybatis에서 제공하는 메소드가 많지만 크게 아래의 메소드를 사용한다.

```
sqlSession.insert("쿼리id", 쿼리에서 빈 값을 채우기 위한 데이터);  
sqlSession.update("쿼리id", 쿼리에서 빈 값을 채우기 위한 데이터);  
sqlSession.delete("쿼리id", 쿼리에서 빈 값을 채우기 위한 데이터);  
sqlSession.selectOne("쿼리id", 쿼리에서 빈 값을 채우기 위한 데이터);  
sqlSession.selectList("쿼리id", 쿼리에서 빈 값을 채우기 위한 데이터);
```

먼저 빨간박스의 메소드명을 보자. 실행할 쿼리가 insert문이면 insert() 메소드를, update문이라면 update()메소드를 실행하면 된다. 즉, 본인이 어떤 쿼리를 실행할 지 쿼리문만 작성할 수 있다면 어떤 메소드를 호출할지 판단하는 것은 쉽게 할 수 있다. Select쿼리의 경우는 호출할수 있는 메소드가 selectOne, selectList 두 가지로 나뉜다. 이 둘을 구분하는 것 역시 쿼리만 작성하면 쉽게 파악할 수 있다.

STU_NUM	STU_NAME	STU_AGE	STU_ADDR	CLASS_CODE
1	1 김자바	20	울산시	1

selectOne() 메소드 호출

STU_NUM	STU_NAME	STU_AGE	STU_ADDR	CLASS_CODE
1	1 김자바	20	울산시	1
2	2 이자바	30	서울시	1
3	3 박자바	25	부산시	1
4	4 김카드	27	울산시	2
5	5 이카드	29	서울시	2
6	6 박카드	35	부산시	2
7	7 김웬디	21	울산시	3

SelectList() 메소드 호출

왼쪽 그림처럼 select쿼리 결과 데이터가 1개의 행 혹은 0개의 행이 나온다면 selectOne을,

Select쿼리 결과 0개가 나올 때도, 1개가 나올 때도, 여러 개가 나올 때도 있을 경우에는

selectList메소드를 호출하면 된다. 반복해서 말하지만, 쿼리를 작성할 수 있다면 사용할 메소드를 선택

하는 것은 어려운 일이 아니다.

자바에서 쿼리를 실행하는 방법(4)

Mybatis 메소드의 첫번째 인자인 쿼리 id 작성 방법

```
<select id="selectBoardList" resultMap="board">
    SELECT BOARD_TITLE
       , BOARD_WRITER
       , TO_CHAR(BOARD_DATE, 'YYYY-MM-DD') AS BOARD_DATE
    FROM SPRING_BOARD
</select>
```

일치해야함

```
public List<BoardVO> selectBoardList(BoardVO boardVO) {
    return sqlSession.selectList("selectBoardList", boardVO);
}
```

```
sqlSession.insert("쿼리id", 쿼리에서 빈 값을 채우기 위한 데이터);
sqlSession.update("쿼리id", 쿼리에서 빈 값을 채우기 위한 데이터);
sqlSession.delete("쿼리id", 쿼리에서 빈 값을 채우기 위한 데이터);
sqlSession.selectOne("쿼리id", 쿼리에서 빈 값을 채우기 위한 데이터);
sqlSession.selectList("쿼리id", 쿼리에서 빈 값을 채우기 위한 데이터);
```

위 메소드에서 첫번째 인자로 받아야하는 쿼리id는 왼쪽 그림하나로 설명이 끝이다.

Mapper파일에서 작성한 쿼리에 id를 주게 되는데, 메소드에서 그 id를 동일하게 주면 쿼리를 실행한다는 의미이다.

자바에서 쿼리를 실행하는 방법(5)

Mybatis 메소드의 두번째 인자 작성 방법

```
sqlSession.insert("쿼리id", 매개변수);  
sqlSession.update("쿼리id", 매개변수);  
sqlSession.delete("쿼리id", 매개변수);  
sqlSession.selectOne("쿼리id", 매개변수);  
sqlSession.selectList("쿼리id", 매개변수);
```

다음은 매개변수를 어떻게 코딩하는지 보겠다.

```
<select id="selectBoard" resultMap="board">  
    SELECT BOARD_TITLE  
        , BOARD_WRITER  
        , BOARD_CONTENT  
        , BOARD_DATE  
        , READ_CNT  
        , BOARD_NUM  
        , BOARD_FILE  
    FROM BOARD  
    WHERE BOARD_NUM = #{boardNum}  
</select>
```

```
<select id="selectBoard" resultMap="board">  
    SELECT BOARD_TITLE  
        , BOARD_CONTENT  
        , BOARD_WRITER  
        , BOARD_DATE  
    FROM SPRING_BOARD  
    WHERE BOARD_TITLE = #{boardTitle}  
</select>
```

```
<update id="updateBoard">  
    UPDATE BOARD  
    SET  
        BOARD_TITLE = #{boardTitle}  
        , BOARD_CONTENT = #{boardContent}  
    WHERE BOARD_NUM = #{boardNum}  
</update>
```

받은 parameter를 쿼리 실행 시 어떻게 전달해 줄지는 마찬가지로 작성된 쿼리문을 보고 판단한다.

위의 세 쿼리는 왼쪽부터 차례대로 1,2,3번이라 지칭하겠다. 먼저 크게 받아와야 되는 parameter 개수가 1개인지, 혹은 2개 이상인지에 따라 parameter 전달 방식이 달라질 수 있다. 또한 parameter가 한 개 일 경우, 해당 parameter가 숫자인지 문자인지에 따라서 다시 구분된다.

정리하자면 parameter를 받아오는 방식은 크게 3가지로 구분되며 parameter가 하나인데 숫자인 경우(1번 쿼리), parameter가 하나인데 문자인 경우(2번 쿼리), parameter가 두개 이상인 경우(3번 쿼리)로 구분이 된다. 이 세가지 경우에 대해 하나씩 살펴보도록 하겠다.

자바에서 쿼리를 실행하는 방법(6)

Mybatis 메소드의 두번째 인자 작성 방법 - Parameter가 1개인 경우

```
<select id="selectBoard" resultMap="board">
  SELECT BOARD_TITLE
    , BOARD_WRITER
    , BOARD_CONTENT
    , BOARD_DATE
    , READ_CNT
    , BOARD_NUM
    , BOARD_FILE
  FROM BOARD
  WHERE BOARD_NUM = #{boardNum}
</select>
```

```
<select id="selectBoard" resultMap="board">
  SELECT BOARD_TITLE
    , BOARD_CONTENT
    , BOARD_WRITER
    , BOARD_DATE
  FROM SPRING_BOARD
  WHERE BOARD_TITLE = #{boardTitle}
</select>
```

지속적으로 말하지만 두번째 인자를 얻어오는 방법도 쿼리문만 작성할 수 있다면 쉽게 알 수 있다.

위의 첫번째 쿼리문에서 parameter로 받아야 하는 값은 #{boardNum} 1개이다.

이렇게 받아야 하는 parameter가 1개인 경우, 그 parameter의 자료형을 파악한다.(boardNum은 숫자형).

두번째 쿼리문 역시 받아야하는 값은 1개이며, 자료형은 문자형이다.

```
public BoardVO selectBoard(int boardNum) {
    return sqlSession.selectOne("selectBoard", boardNum);
}
```

Id가 selectBoard인 쿼리를 실행할 때 빈값은 boardNum으로 채우라는 의미이다.
boardNum이 숫자이기 때문에 매개변수를 int로 준 것이다.

```
public BoardVO selectBoard(String boardTitle) {
    return sqlSession.selectOne("selectBoard", boardTitle);
}
```

Id가 selectBoard인 쿼리를 실행할 때 빈값은 boardTitle로 채우라는 의미이다.
boardTitle이 문자이기 때문에 매개변수를 String로 준 것이다.

자바에서 쿼리를 실행하는 방법(7)

Mybatis 메소드의 두번째 인자 작성 방법 - Parameter가 여러개인 경우

```
<update id="updateBoard">
  UPDATE BOARD
  SET
  BOARD_TITLE = #{boardTitle}
  , BOARD_CONTENT = #{boardContent}
  WHERE BOARD_NUM = #{boardNum}
</update>
```

해당 쿼리문의 경우 parameter로 받아야 하는 값이 총 3개이다. 이렇게 받아야 하는 parameter가 여러 개일 때는 매개변수로 VO객체를 사용한다.

VO객체 하나에 받아야 하는 boardTitle, boardContent, boardNum값을 한번에 저장할 수 있기 때문이다. 그리고 VO클래스에는 이름이 같은 변수와 GETTER와 SETTER가 반드시 있어야 한다.

```
public int updateBoard(BoardVO boardVO) {
    return sqlSession.update("updateBoard", boardVO);
}
```

Id가 updateBoard인 쿼리를 실행할 때 빈값은 boardVO으로 채우라는 의미이다.

자바에서 쿼리를 실행하는 방법(8)

최종적으로 정리하자면, 자바에서 쿼리를 실행시킬 수 있으려면 아래의 능력이 충족되어야 한다.

첫번째, 먼저 쿼리를 작성할 수 있어야 한다.

수업시간에 실습을 통해 작성하라는 쿼리문은 어려운 내용이 아니다. 기본 쿼리문이기 때문에 하루만 투자해서 집중해서 책을 본다면 무난하게 작성할 수 있는 레벨이다. 어려운 쿼리문은 학습을 통해 하나씩 배우고 있으니, 기본 쿼리문을 작성하지 못하는 것은 본인이 노력해야 한다. 아직 게시판 구현을 위한 조회, 수정, 삭제, 삽입 등의 쿼리 작성이 어렵다면 하루만 투자해서 책 보자!

두번째, Mybatis의 어떤 메소드를 호출한지 판단한다.

4번 슬라이드 참조

세번째, 작성한 쿼리를 통해 자바에서 받아야 하는 parameter가 무엇인지 파악할 수 있어야 한다.

이는 쿼리문만 작성할 수 있다면 누구나가 파악할 수 있다. 다시 한번 말한다. 받아와야하는 parameter는 #{ }로 표현된 것이다.

네번째, parameter를 파악했다면, 실제 해당하는 parameter를 쿼리문에 어떻게 전달해야 하는지 알아야 한다.

6 ~ 8 슬라이드 참조

쿼리 결과를 자바로 가져오는 방법(1)

앞선 슬라이드를 통해 자바에서 쿼리를 실행시키는 방법에 대해 학습하였다. 이번 슬라이드부터는 실행한 쿼리 결과를 자바로 가져오는 방법에 대해 학습한다. 아래 그림에서 빨간박스에 해당하는 부분은 자바에서 쿼리를 실행시키기 위해 작성해야하는 코드 부분이다. 그리고 파란색 박스가 쿼리의 결과를 가져오기 위해 작성해야 하는 부분이다.

```
public int updateBoard(BoardVO boardVO) {  
    return sqlSession.update("updateBoard", boardVO);  
}
```

쿼리를 실행시킬 때 작성해야 하는 코드 부분(메소드 선택과 매개변수 선택)

쿼리 실행 결과를 가져오기 위해 작성한 코드

다시 말해 service파일에서 메소드의 리턴값을 어떤 자료형으로 결정하냐에 따라 쿼리 실행 결과를 자바에서 필요한 형태로 받아 올 수 있다.

리턴 타입은 무수히 많은 타입으로 받아 올 수 있지만 가장 기본이 되는 4가지의 리턴값(String, int, VO, List<VO>)에 대해서 알아보도록 하겠다.

쿼리 결과를 자바로 가져오는 방법(2)

메소드의 리턴타입을 결정할때에도 가장 먼저 생각할 것은 쿼리문이다. 쿼리문만 작성한다면 리턴값을 결정하는 것은 어렵지 않다. 아래는 Sql Developer에서 실행한 쿼리의 결과를 나타낸 것이다.

```
75 INSERT INTO SPRING_BOARD VALUES ('제목', '내용', '작성자', SYSDATE);
76
```

스크립트 출력 x

작업이 완료되었습니다. (0.052초)

1 행 이(가) 삽입되었습니다.

INSERT 쿼리 실행

```
77 UPDATE SPRING_BOARD
78 SET BOARD_WRITER = '작성자2'
79 WHERE BOARD_TITLE = '제목';
```

스크립트 출력 x

작업이 완료되었습니다. (0.029초)

1 행 이(가) 업데이트되었습니다.

UPDATE 쿼리 실행

```
81 DELETE SPRING_BOARD
82 WHERE BOARD_TITLE = '제목';
```

스크립트 출력 x

작업이 완료되었습니다. (0.029초)

1 행 이(가) 삭제되었습니다.

DELETE 쿼리 실행

```
84 SELECT BOARD_TITLE, BOARD_CONTENT FROM SPRING_BOARD
85 WHERE BOARD_TITLE = '제목';
```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 1(0.002초)

BOARD_TITLE	BOARD_CONTENT
제목	내용

SELECT 쿼리 실행-1

```
84 SELECT BOARD_TITLE, BOARD_CONTENT FROM SPRING_BOARD;
```

스크립트 출력 x | 질의 결과 x

SQL | 인출된 모든 행: 5(0.001초)

BOARD_TITLE	BOARD_CONTENT
속드송기드기	올로올로
안녕하세요	안녕
11111111	44444
Java	fddfd
제목	내용

SELECT 쿼리 실행-2

쿼리 결과를 자바로 가져오는 방법(3)

INSERT, UPDATE, DELETE 쿼리문의 리턴타입

```
75 INSERT INTO SPRING_BOARD VALUES ('제목', '내용', '작성자', SYSDATE);
76
```

스크립트 출력 x

작업이 완료되었습니다. (0.052초)

1 행 이(가) 삽입되었습니다.

INSERT 쿼리 실행

```
77 UPDATE SPRING_BOARD
78 SET BOARD_WRITER = '작성자2'
79 WHERE BOARD_TITLE = '제목';
```

스크립트 출력 x

작업이 완료되었습니다. (0.029초)

1 행 이(가) 업데이트되었습니다.

UPDATE 쿼리 실행

```
81 DELETE SPRING_BOARD
82 WHERE BOARD_TITLE = '제목';
```

스크립트 출력 x

작업이 완료되었습니다. (0.029초)

1 행 이(가) 삭제되었습니다.

DELETE 쿼리 실행

INSERT, UPDATE, DELETE 쿼리문을 SQL Developer에서 실행하면 위의 이미지와 같은 결과가 나타난다.

‘1행이 삽입되었습니다’, ‘1행이 업데이트 되었습니다.’ 등 몇 개의 행이 변화되었다는 결과를 보여준다. 3개의 데이터가 동시에 들어가면 ‘3개의 행이 삽입되었습니다.’라는 문구가 나오고, 0개의 행이 삭제되면 ‘0개의 행이 삭제되었습니다.’라는 문구가 뜬다.

정리하자면, INSERT, UPDATE, DELETE 쿼리문의 실행결과 **몇 개의 행이 변화되었는지에 대한 결과가 중요하기** 때문에 리턴값은 int형이 된다.

예를 들어, 실행 결과 ‘3개의 행이 삽입되었습니다.’ 라는 문구가 나타나는 쿼리문의 리턴값을 int로 하면 int에는 3이라는 숫자가 반환된다.

즉, INSERT, UPDATE, DELETE 쿼리문은 메소드 작성 시 리턴값을 int로 표현하면 아무 문제 없이 실행 결과를 받아 올 수 있다.

쿼리 결과를 자바로 가져오는 방법(4)

SELECT 쿼리문의 리턴타입

```
84 SELECT BOARD_TITLE FROM SPRING_BOARD
85 WHERE BOARD_TITLE = '제목';
86
```



BOARD_TITLE
제목

쿼리 실행 결과 '제목'이라는 데이터 한 개가 나왔다. '제목'이라는 내용은 하나의 문자열이기 때문에 String 타입으로 데이터를 받을 수 있다. 그렇기 때문에 이와 같은 경우에는 리턴 타입을 String으로 할 수 있다. 추가적으로 만약 결과 데이터가 '10'과 같은 숫자 데이터 1개가 나온다면 리턴 타입을 int로 할 수 있다.

SELECT 쿼리 실행

```
-1
84 SELECT BOARD_TITLE, BOARD_CONTENT FROM SPRING_BOARD
85 WHERE BOARD_TITLE = '제목';|
```



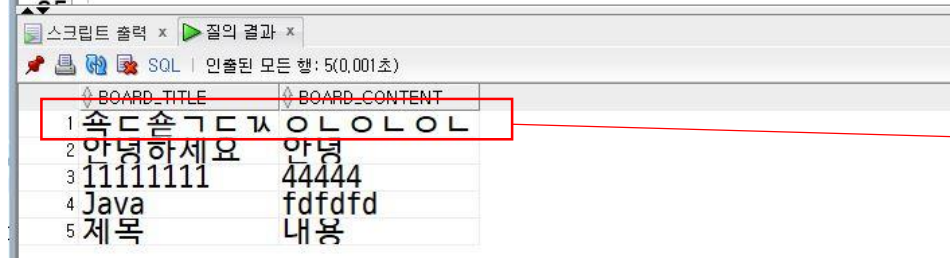
BOARD_TITLE	BOARD_CONTENT
제목	내용

SELECT 쿼리 실행-2

데이터 조회 결과 데이터 한 줄(1행)의 데이터는 VO객체에 저장할 수 있다.

해당 쿼리문의 결과 '제목', '내용'이라는 문자열 데이터가 두 개 나왔다. 문자열 두 개는 String이나 int 타입으로 받을 수 없다. 그렇기 때문에 해당 데이터를 받을 수 있는 자료형을 별도로 개발자가 작성해야 하는데 이것이 VO객체이다. VO객체에는 boardTitle과 boardContent의 변수가 존재하고 해당 변수에 각각의 데이터를 넣을 수 있다.

```
84 SELECT BOARD_TITLE, BOARD_CONTENT FROM SPRING_BOARD;
```



BOARD_TITLE	BOARD_CONTENT
속도속가드기	오로오로
안녕하세요	안녕
111111111	44444
Java	fdfdfd
제목	내용

SELECT 쿼리 실행-3

데이터 조회 결과 데이터 한 줄(1행)은 VO객체와 같다. 왼쪽의 쿼리 결과 결과 행이 5줄이다. 한줄 한줄이 VO객체이기 때문에 지금같은 경우, VO객체가 5개 있어야 모든 데이터를 저장할 있다. 따라서, VO객체를 여러 개 담을 수 있는 자료형인 List<VO>자료형을 사용하게 된다.