Spring Boot Part6

Mybatis 데이터 바인딩 이해하기
Mybatis if문
Mybatis foreach문
Mybatis 조인 데이터 조회하기

```xml
<insert id="insertBoard">
    INSERT INTO BASIC_BOARD (
        TITLE
        , WRITER
        , CONTENT
    ) VALUES (
        #{title}
        , #{writer}
        , #{content}
    )
</insert>
```

Mybatis Framework에서 쿼리문은 xml 파일에서 작성한다. 이때 쿼리에서 채울 값을 xml 파일에서 왼쪽과 같이 작성하였다. #{}에 작성하는 내용은 쿼리 실행을 위해 채워줘야하는 데이터이며, #{} 안의 내용은 DTO 클래스에 선언한 변수명과 동일하게 작성하였다. 이렇게 #{} 안의 내용을 DTO 클래스 변수명과 동일하게 작성하는 것은 Mybatis가 데이터를 바인딩을 하는 방식 때문이다. 아래 코드는 왼쪽의 쿼리문에 빈 값을 채워 쿼리를 실행하는 메서드를 작성한 것이다.

```java
//게시글 등록 쿼리 실행 메서드
public int insertBoard(BoardDTO boardDTO);
```

위 메서드에서 매개변수에 작성한 BoardDTO 클래스의 객체는 쿼리의 빈 값을 채워주는 역할을 한다. BoardDTO 객체가 쿼리의 빈 값을 채울 때는 getter를 호출하여 빈 값을 채운다. 쿼리문에 #{title}은 boardDTO.getTitle() 메서드가 호출되어 채워지며,
#{writer}는 boardDTO.getWriter() 메서드가 호출되어 데이터를 채우는 것이다.
만약, #{TITLE}이라고 쿼리문에 작성했다면 boardDTO.getTITLE() 메서드가 호출될 것이다. BoardDTO 클래스에는 getTITLE() 이라는 메서드가 존재하지 않기 때문에 실행 시 오류가 발생한다. 이렇듯 #{}안에 작성한 변수의 gettet를 호출하기 때문에, #{} 안의 내용은 대문자로 작성하지 않고, DTO 클래스의 변수명과 반드시 일치시켜 작성해야 하는 것이다.

또 다른 쿼리문과 쿼리를 실행시킬 메서드의 정의를 보자.

```xml
<select id="getBoard" resultType="BoardDTO">
    SELECT *
    FROM BASIC_BOARD
    WHERE BOARD_NUM = #{boardNum}
</select>
```

실행할 쿼리

```java
//게시글 상세 조회 쿼리 실행 메서드
public BoardDTO getBoard(int boardNum);
```

쿼리 실행 메서드

앞선 슬라이드에서 이야기한 것 처럼 위 쿼리문의 #{boardNum}도 getter를 호출할까? 그렇지 않다.

위 쿼리문의 빈 값을 채우기 위해 전달된 데이터는 int boardNum이다. int형 데이터에는 애초에 getter가 존재할 수 없다.

결국, 위 쿼리문의 빈 값은 getter를 호출하는 것이 아니라, 단순히 빈 값 하나를 채우는 것에 목적이 있다. 매개변수로 전달되는 데이터로 빈 값을 채우겠다는 의미이지, 이름은 중요하지 않다는 것이다. 실제로 #{boardNum}이라고 작성된 부분에 아무 글자나 작성해보면 모두 정상적으로 실행되는 것을 알 수 있다. 그럼에도 쿼리문에 #{boardNum}이라고 작성하는 이유는 글 번호가 전달된다는 의미를 코드에 녹여내기 위해서이다.

정석은 #{_parameter}를 작성하는 것이다.

정리

쿼리의 빈 값을 DTO 클래스의 객체로 채울 때는 #{}의 내용은 getter 호출로 해석한다.

쿼리의 빈 값이 하나이며, 그 값을 int 혹은 String으로 채울 때는 #{}에 작성한 글자는 중요하지 않다. 그냥 전달되는 데이터 하나로 빈 값 하나만 채우면 그만이다. 그렇기에 아무렇게나 작성하면 되지만 통상적으로 전달되는 변수명을 그대로 작성하거나 _parameter로 작성한다.

지금껏 우리는 쿼리문에서 채워야 할 값을 #{} 안에 작성했지만, 사실 Mybatis에서 쿼리문의 빈 값을 채울 때에는 #{}, ${} 두 가지 방법을 제공한다.

```
<select id="getBoard" resultType="BoardDTO">
    SELECT *
    FROM BASIC_BOARD
    WHERE BOARD_NUM = #{boardNum}
</select>
```

#{}을 이용한 데이터 바인딩

```
<select id="getBoard" resultType="BoardDTO">
    SELECT *
    FROM BASIC_BOARD
    WHERE BOARD_NUM = ${boardNum}
</select>
```

${}을 이용한 데이터 바인딩

#{}과 ${}는 해석에서 차이점이 발생한다. #{}로 작성된 경우에는 데이터를 홀따옴표에 감싸서 표현하고, ${}는 홀따옴표로 감싸지 않는다.

위 쿼리문에서 WHERE절에 #{}을 사용했을 때 WHERE BOARD_NUM = '10' 으로 해석되며, ${}를 사용하면 WHERE BOARD_NUM = 10으로 해석되는 것이다. BOARD_NUM 값이 INT형이기 때문에 홀따옴표를 붙이면 쿼리가 잘못 작성된 것 아니냐고 생각할 수 있지만 그렇지는 않다. 데이터베이스가 알아서 형변환을 해주기 때문에 WHERE BOARD_NUM = '10'으로 작성해도 '10'을 숫자 10으로 변환해 WHERE BOARD_NUM = 10의 쿼리를 실행해 버린다. 그렇기 때문에 홈따옴표의 유무는 숫자와 문자 데이터를 구분할 때 사용하는 것이 아니다. 바로 채워야 하는 정보가 일반 데이터인지, 혹은 컬럼명인지에 따라 달라진다.

다음 쿼리가 #{}, ${}의 차이점을 잘 나타내는 쿼리문이다.

```
<select id="getBoard" resultType="BoardDTO">
    SELECT *
    FROM BASIC_BOARD
    WHERE #{data} = '김자바'
</select>
```

#{}을 이용한 데이터 바인딩

```
<select id="getBoard" resultType="BoardDTO">
    SELECT *
    FROM BASIC_BOARD
    WHERE ${data} = '김자바'
</select>
```

${}을 이용한 데이터 바인딩

위 쿼리에서 만약 data라는 곳에 TITLE이라는 값이 전달된다면 #{}을 사용하면 다음과 같이 해석된다.

SELECT * FROM BASIC_BOARD WHERE 'TITLE' = '김자바'

반대로 ${}을 사용한 결과의 쿼리문은 다음과 같다.

SELECT * FROM BASIC_BOARD WHERE TITLE = '김자바'

위 두 쿼리문은 완전히 다른 결과를 갖는 쿼리문이다. #{}을 사용한 쿼리문은 TITLE이라는 문자열과 '김자바'라는 문자열이 같은 데이터만 조회한다는 의미인데, 이러한 데이터는 존재하지 않으므로 조회 데이터가 0건이다. 반면, ${}을 사용한 쿼리문은 TITLE컬럼의 값이 '김자바' 문자열과 같은 데이터를 조회하기 때문에 데이터가 조회된다. 결국 일반적인 데이터를 태워줄때는 #{}을 사용하면 되는 것이고, 컬럼명을 채워줘야 하다면 홀따옴표로 감싸지 않게 ${}를 사용해야 한다.

Mybatis는 동적 쿼리문 작성을 지원하는 여러 문법이 존재한다. 가장 많이 사용하는 것이 쿼리문의 if문 사용이다.

다음 예시는 쿼리문에서 if문을 사용한 예시이다.

```
<select id="getBoard" resultType="BoardDTO">
    SELECT *
    FROM BASIC_BOARD
    WHERE TITLE = #{title}
    <if test='writer != null and !writer.equals("")'>
        AND WRITER = #{writer}
    </if>
</select>
```

실행할 쿼리

```
public List<BoardDTO> getBoard(BoardDTO boardDTO);
```

쿼리 실행 메서드

<if test="">안의 쿼리문은 test="" 안에 작성한 내용이 true로 해석될 때 실행된다.

test=" 안에 작성한 내용은 무조건 getter를 호출하여 해석한다.(해당 예시에서는 boardDTO.getWriter())

위 if문 안의 쿼리문은 boardDTO.getWriter()의 결과로 받은 데이터가 null이 아니면서 빈 문자로 아닐때 실행하게 된다.

이때, 주의사항은 test문장에서 반드시 쌍따옴표가 아닌 홀따옴표를 사용해야 한다는 점이다. 그렇지 않으면 equals("") 에 작성한 쌍따옴표 때문에 제대로 해석되지 않는다.

다음의 예시를 보자.

```
<select id="getBoard" resultType="BoardDTO">
    SELECT *
    FROM BASIC_BOARD
    <if test='title != null and !title.equals("")'>
        WHERE TITLE = #{title}
    </if>
    <if test='writer != null and !writer.equals("")'>
        AND WRITER = #{writer}
    </if>
</select>
```

실행할 쿼리

```
public List<BoardDTO> getBoard(BoardDTO boardDTO);
```

쿼리 실행 메서드

위 쿼리처럼 if문은 필요시 여러번 사용할 수 있다. 이런 쿼리문에서 첫번째 if문의 해석은 false가 되고, 두번째 if문의 해석은 true가 되는 경우를 생각해 보면 최종 쿼리문은 다음과 같을 것이다.

SELECT * FROM BASIC_BOARD

AND WRITER = #{writer}

이러한 쿼리는 문법에 맞지 않다. 이렇게 if문을 사용하면 WHERE절과 AND절 사용에 각별히 유의해야 한다. 이를 해결하기 위해서는<WHERE>을 사용한다.

앞선 슬라이드에서의 문제점을 해결한 쿼리문이 아래의 쿼리문이다.

```
<select id="getBoard" resultType="BoardDTO">
    SELECT *
    FROM BASIC_BOARD
    <where>
        <if test='title != null and !title.equals("")'>
            AND TITLE = #{title}
        </if>
        <if test='writer != null and !writer.equals("")'>
            AND WRITER = #{writer}
        </if>
    </where>
</select>
```

실행할 쿼리

<WHERE> 태그 안에 작성된 쿼리문을 모두 AND로 시작하면 최초로 조건이 맞아 실행되는 쿼리문의 AND를 WHERE절로 치환시켜준다.

만약, 두 if 조건이 모두 true라면,

WHERE TITLE = #{title}

AND WRITER = #{writer}

로 해석하며, 두번째 if문의 조건만 true라면,

WHERE WRITER = #{writer} 로 해석해준다.

Mybatis는 자바에서의 if문을 대신할 수 있는 <if></if> 태그는 제공하지만, 자바의 eles 문을 대체하는 태그는 제공하지 않는다.

else문은 <if></if> 태그 혹은 <choose> 태그를 사용하여 해결한다.

만약 다음과 같은 쿼리문과 메서드에 if문을 사용하면 어떻게 될까.

```xml
<select id="getBoard" resultType="BoardDTO">
    SELECT *
    FROM BASIC_BOARD
    <if test="boardNum != 0">
        WHERE BOARD_NUM = #{boardNum}
    </if>
</select>
```

실행할 쿼리

```
//게시글 상세 조회 쿼리 실행 메서드
public BoardDTO getBoard(int boardNum);
```

쿼리 실행 메서드

앞선 5번 슬라이드에서 <if> 태그의 test 안에 작성된 내용은 getter를 호출하여 해석한다고 하였다.

하지만 이번 쿼리의 빈 값을 채우기 위해서 전달된 데이터는 int형 데이터이다. int형 데이터는 getter가 없기 때문에 위 쿼리문에서 작성한

<if test="boardNum != 0"> 에서 boardNum은 getter로 해석되면 오류가 발생한다. 이렇게 쿼리의 빈 값을 채우기 위해 전달되는 데이터가 int 혹은

String 데이터일 경우에는 반드시 _parameter를 사용해야 한다. 위 쿼리문을 올바르게 수정하면 다음과 같은 코드가 된다.

```xml
<select id="getBoard" resultType="BoardDTO">
    SELECT *
    FROM BASIC_BOARD
    <if test="_parameter != 0">
        WHERE BOARD_NUM = #{boardNum}
    </if>
</select>
```