2024/10/10 6일차

마이바티스

https://blog.mybatis.org/ → **Products**

ORM framework

객체와테이블매핑

SQL Mapper framework

객체와SQL을 매핑

SQL실행한값을 객체에 담고 다른 객체에 전달해줌

- **SQL이 실행하는데 필요한 값(파라미터)을 어느객체에 담아서 전달할건지 알려줘야함**
  *** 핵심***

mybatis

- **SQL Mapper framwork**이다.
- mybatis를 사용하면 JDBC의 반복적인 코드를 제거할 수 있다.
  - Connection, PreparedStatement객체를 획득하고, 해제하는 코드
  - SQL의 바인딩변수(?)에 파라미터값을 바인딩시키는 코드
  - SQL 실행결과로 획득된 조회결과를 ResultSet에서 값을 꺼내서 객체에 담는 코드
- **주요 구성요소**
  - mybatis 설정 파일 (Configuration 파일 (mybatis-config.xml))

        - 데이터 베이스 연결정보
        - 매퍼파일 정보를 포함
          * spring boot를 사용하는 경우 생성할 필요가 없다.
    - **매퍼파일**
        - 자바객체와 매핑된 SQL 구문이 정의된 파일
        - 데이터베이스의 테이블마다 매퍼파일 하나씩 필요하다.
        - XML 형식의 파일이다.
        - <insert>, <select> , <update>, <delete> 등의 태그를 이용해서 SQL을 작성한다.
    - **매퍼인터페이스**
        - 테이블에 대한 CRUD 작업을 추상메소드로 정의하는 인터페이스다.
        - 데이터베이스의 테이블마다 매퍼인터페이스가 하나씩 필요하다.
        - 매퍼파일에서 작성하는 SQL은 매퍼인터페이스가 정의된 추상 메소드와 대응된다.
    - **매퍼인스턴스**
        - 매퍼인터페이스 구현체다
        - mybatis는 인터페이스의 추상메소드오 ㅏ매핑된 SQL 구문을 실행하도록 구현체를 생성한다.
        - spring과 연동되면 mybatis가 매퍼인터페이스를 생성하고, 스프링의 빈으로 등록시킨다.

        > 💡 개발자는 매퍼파일, 매퍼인터페이스를 정의한다.

        인터페이스 → 매퍼파일 |
        인터페이스 → 매퍼파일 | → 설정파일 | → 마이바티스 → 매퍼 인스턴스 생성
        DB연결 |

- 주요 API
    - SqlSessionFactoryBuilder (마이바티스 단독으로 사용시 등장함)

        - mybatis 설정정보를 제공받아서 SqlSessionFactory객체를 생성하는 객체다.
        - SqlSessionFactiory 생성이 완료되면 더 이상 필요없는객체다.
    - SqlSessionFactory
        - SqlSession객체를 제공하는 팩토리 객체다.
        - 애플리케이션이 실행되는 동안 오직 하나만 생성해서 계속 사용하는 객체다.
    - SqlSession
        - SQL 구문을 실질적인 데이터 베이스 엑세스 작업을 사행하는 객체다.
        - 데이터베이스 엑세스 작업을 수행할 때마다 SqlSessionFactory로부터 SqlSession객체를 획득해서 사용하고,
          엑세스 작업이 완료되면 폐기한다.

          * 위의 객체들은 mybatis 단독으로 애플리케이션에서 사용될 때 필요하다
          * spring 혹은 spring boot와 연동해서 mybatis를 사용하면 결코 사용할 일이 없다.
- mybatis의 특징
    - SQL과 자바객체를 매핑하기만 하면 된다. (개발자가 데이터베이스 엑세스 작업을 수행하기 위해서 최소한으로 할 일)
        - 위의 작업만 작성하면 mybatis가 자동으로 SQL과 자바객체를 연결해서 SQL 실행에 필요한 모든 작업을 자동으로 수행한다.
        - SQL이 자바코드와 분리되어 별도의 XML 파일로 관리되기 때문에 SQL을 유지보수하기 편하다.
    - 다이나믹 SQL 생성 기능을 지원한다.
        - SQL에 전달된 파라미터값에 따라서 서로 다른 SQL 문으로 동적으로 생성해내는 기능을 제공한다.
        - SQL에서 대부분의 코드는 동일하고 조회조건만 다른 SQL이 사용되는 경우가 많은데 이를 때
          다이나믹 SQL 생성 기능을 이용하면 조회조건이 다른 SQL 구문을 여러개 정의할 필요없이
          동적으로 파라미터값에 따라서 적절한 조회조건이 적용된 SQL 구문을 생성할 수 있다.
- mybatie의 장점과 단점

    - 장점
        - mybatis는 SQL에 대한 직접적인 제어를 제공한다.
          (개발자가 직접 SQL을 작성하기 때문에 복잡한 쿼리, 특정베이스에 최적화된 쿼리 작성이 가능하다.)
        - JPA에 비해서 학습이 용이하다.
    - 단점
        - 모든 테이블에 대해서 CRUD 같은 기본적인 코드를 계속 만들어야 한다.
        - 데이터베이스에 종속적이다.
          * SQL을 직접 작성하기 때문에 데이터베이스가 변경되면 SQL을 수정해야 한다.

- mybatis를 이용한 데이터베이스 엑세스 작업 순서
    - 1. 테이블 생성
    - 2. 도메인 객체 설계 및 클래스 작성
      User.java, Book.java, Order.java, Review.java
    - 3. 매퍼 인터페이스 작성

```
interface UserMapper{
  void insertUser( User user);
  void deleteUserByNo( int no);
List<User> searchUserByName(String name);
User getUserByNo(int no);
…
}
interface BookMapper{
  List<Book> getBooks(int catNo, int begin, int end);
  Book getBookByNo(int no)
  …
}
```

    - 4. **매퍼 파일 작성** (**xml을 잘 못 작성할 경우 전체 코드가 작동을안함**)

```
   UserMapper.xml
         <insert id="insertUser" parameterType="vo.Use
```

```
r">
            insert into users
            ...
        </insert>
        <delete id="deleteUserByNo" parameterType="in
t">
            delete from users
            ...
        </delete>

    BookMapper.xml
        <select id="getBooks" resultType="vo.Book"
            select book_no as no
            ...
        </select>
```

- 5. UserService,BookService와 같은 서비스 클래스에서 UserMapper, BookMapper를 DI로
  주입받아서 실제 업무로직 수행에 필요한 데이터베이스 엑세스를 수행하기

```
@Service
class UserService{
    @Autowired
    private UserMapper userMapper;
    @Autowired
    private OrderMapper orderMapper;
    @Autowired
    private PoinHistoryMapper pointHistoryMapper;

    public void registerUser(User user) {
        User savedUser = userMapper.getUserById(user.get
id());
            if(savedUser != null){
                    throw new RuntimeException("아이디 중
복");
            }
            ...
            userMapper.insertUser(user);
```

```
    }
}
* spring, spring-boot와 mybatis를 연동해서 사용하면 mybatis가 매퍼 인터페이스와 연관된 매퍼파일의 SQL 구문을
    실행하는 매퍼 인스턴스(매퍼 인터페이스를 구현한 객체)
  매퍼 인스턴스( 매퍼 인터페이스를 구현한 객체)를 동적으로 생성하고, 스프링 컨테이너의 빈으로 등록시킨다.

* 테이블에 대한 데이터베이스 엑세스 작업이 필요한 객체는
    해당 매퍼 인터페이스 타입의 멤버변수를 정의하고 @Autowired 어노테이션으로
    객체를 자동으로 주입받아서 사용할 수 있다.
```

```
마이바티스 프레임워크
============================================================
============================================================
=============

    ============================================================
====
            매퍼 인터페이스
            public interface UserMapper {
                void insertUser(User user) ; // 신규 사용자 저장
                User getUserByNo(int no); // 사용자 조회
                void deleteAllUsers{}; // 전체 사용자 삭제
                ... 기타등등
            }

    ============================================================
====
```

```
    ============================================================
====
            매퍼파일
            <insert id="insertUser" parameter type="User">
                insert into users(user_no, user_id, user_na
me, ...)
                valuer(user_seq.nextaval, #{id}, #{nam
e}...)
            </insert>

            <select id="getUserByNo" parameterType="int" re
sultType="User">
                select user_no as no, user_id as id ...
                from users
                where user_no =#{no}
            </select>
    ============================================================
====


============================================================
============================================================
=============



마이바티스 설정파일에 등록
     |
마이 바티스가 자동구현
     |
============================================================
============================

마이바티스가 구현한 것

public class UserMapperProxy implements UserMapper{
```

```
    ...
    ...
    ...
    @Override
    pulbic void insertUser(User user){
        sqlSession.insert("insertUser",user); 핵심객체
    }

    @Override
    public User getUserByNo(int no) {
        ...
        User x = sqlSession.selectOne("getUserByNo",no);
        ...

        return x;
    }
}

============================================================
============================

    select만 반환값이 있음 다른건 다 없음
    get,find search등으로 이름 명확하게 하기
    우리는 인터페이스와 매퍼파일을 작성한다.
```

선생님이 그려주신 이미지 넣기 스프링이 Uservice, UserMapperProxy 빈으로 등록시킴

UserService {

@Autowired UserMapper userMapper;

}

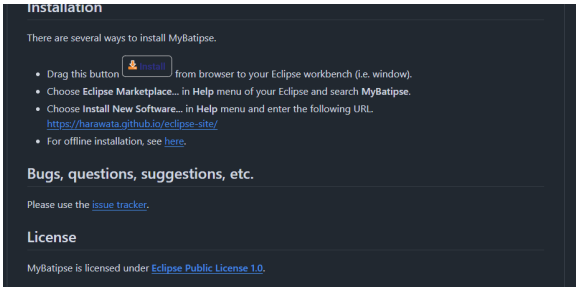
만들어보기

```
package vo;

publ  ic class Job {

    private String id;
    private String title;
    private int minSalary;
    private int maxSalary;

    public Job() {}
    //기본생성자가 무조건 있어함 다른 생성자를 정의하면 기본생성자가 생성되지않기때문에 꼭 기본생성자 정의하기

    public String getId() {
        return id;
    }


    public void setId(String id) {
        this.id = id;
    }


    public String getTitle() {
        return title;
    }


    public void setTitle(String title) {
        this.title = title;
    }


    public int getMinSalary() {
        return minSalary;
    }
```

```
    public void setMinSalary(int minSalary) {
        this.minSalary = minSalary;
    }


    public int getMaxSalary() {
        return maxSalary;
    }


    public void setMaxSalary(int maxSalary) {
        this.maxSalary = maxSalary;
    }


    @Override
    public String toString() {
        return "Job [id=" + id + ", title=" + title + ", mi
nSalary=" + minSalary + ", maxSalary=" + maxSalary + "]";
    }


}
```

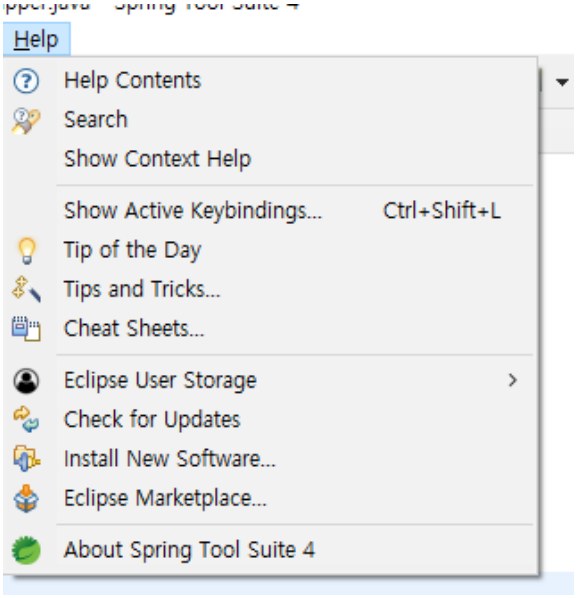
오후 수업

플러그인 넣기



링크 복사해서

XML



install new Software 클릭후 Add 클릭 아래에 url 집어넣기 플러그인 설치하는거임

```
<?xml version="버전" encoding="인코딩방식" standalone="yes 아니면 no"?>
xml 문서는 첫줄 무조건 ?xml 이 와야함 주석조차도 안됨 오류남
================================================================================

<?xml version="1.0" encoding="UTF-8"?> <- XML 문서 선언


<!DOCTYPE mapper PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd"> <- XML문서타입 선언(DTD 선언)
        xml 문서의 형식을정의(dtd,xsd가있다.)
            이건 DTD
            1.공개된 작성규칙이다. PUBLIC
            2.비공인단체 -
            3.작성한 기관 mybatis.org
            4.작성규칙의 이름 DTD Mapper 3.0
            5.영어로 작성함 EN
```

```
<mapper namespace="mybatis.mappers.JobMapper">
</mapper>  <- XML 문서의 루트 엘리먼트

head  : 1번
head+ : 1번 이상
head? : 0또는 1번
head* : 0번 아니면 여러번
```

- **XML 문서 선언**
  - 이 문서가 XML 문서임을 나타내는 선언이다.
  - 모든 XML문서는 해당 문서의 첫번째 줄에 XML 문서 선언이 정의되어야 한다.
- **XML 문서 타입선언**
  - 이 XML 문서의 작성규칙이 정의된 DTD를 나타낸다.
  - 위의 내용은 <!DOCTYPE> XML 문서타입을 정의한다.
    mapper : 이 XML 문서의 루트엘리먼트를 정의한다.
    public : 여기서 사용한 DTD는 공개된 DTD임을 나타낸다.
    “-//mybatis.org//DTD Mapper 3.0//EN”:비공인 국제단체인 mybatis.org에서 DTD Mapper 3.0이라는 이름을 가진 작성규칙 영어로 작성하였다.
    "http://mybatis.org/dtd/mybatis-3-mapper.dtd" : 작성규칙이 정의된 DTD파일의 경로다.
- **XML 문서의 루트 엘리먼트**
  - XML 문서의 루트 엘리먼트다. 모든 엘리먼트의 이 루트엘리먼트 내부에 정의된다.

```
package kr.co.jhta.mapper;

public interface JobMapper {
    void insertJob(Job job);
    Job getJobById(String id);
    List<Job>getAllJobs();
}
```

```
<mapper namespace="kr.co.jhta.mapper.JobMapper> <- 이름 반드
시 똑같이 넣어야함 다르게 할 시 구현체를 안 만들어줌

<insert id= "insertJob" parameterType="kr.co.jhta.vo.Job">

</insert>

<select id="getJobById" parameterType="string" resultType
="kr.co.jhta.vo.Job">

</select>


<select id="getAllJobs" resultType="kr.co.jhta.vo.Job">

</select>

</mapper>
```

- **1. namespace의 속성 정의**
  - <mapper namespace="**이 매퍼파일과 관련된 매퍼인터페이스의 전체패키지경로 및 인터페이스 이름**"> *** 중요함 ***
- **2. id 속성의 정의**
  - **매퍼 인터페이스의 추상 메소드 이름**이다.

    *** 중복값을 허용하지 않는다.**
- 3. **parameterType과 resultType**
  - **parameterType과 resultType**
    - SQL 구문의 ? 에 치환될 값을 담고 있는 클래스의 이름 혹은 기본 자료형의 이름이다.
      매퍼인터페이스의 매개변수 타입과 일치한다.
      기본자료형타입과 String타입 및 자주 사용되는 자바의 몇몇 클래스는 별도의

별칭이 정해져 있다.
매개변수가 없으면 생략한다.

- **resultType**
  - select구문의 실행결과를 저장하는 클래스의 이름 혹은 기본자료형의 이름이다.
    <insert>,<update>,<delete> 태그에서는 사용할 수 없다.
    매퍼 인터페이스의 반환타입과 일치한다. 단 List<Job>과 같이 반환타입이 콜렉션일 때는 Job을 적는다.

insert 구문 작성하기

Job객체를 생성하고

Job객체의

id,title... 에 값을 담아서 전달하자

⇒ SQL

insert into jobs

(job_id,job_title,min_salary,max_salary)
values
(?,?,??)

우리가 작성해야할것들

- 매퍼 인터페이스에 추상 메소드 정의하기
  - void insertJob(Job job);
- 매퍼파일 작성하기
  - <insert id="insertJob" parameterType="kr.co.jhta.vo.Job">
    (JOB_ID, JOB_TITEL, MIN_SALARY, MAX SALARY)
    VALUES
    (#{
    id},#{title},#{minSalary},#{maxSalary}) JOB객체의 멤버변수 이름
    </insert>

mybatis-spring 연동하기

- 1. 라이브러리 추가 : mybatis와 mybais-spring 라이브러리를 pom.xml에 추가

```
<!-- mabatis 라이브러리-->
     <dependency>
          <groupId>org.mybatis</groupId>
          <artifactId>mybatis</artifactId>
          <version>3.5.16</version>
     </dependency>
        <!-- mybatis와 스프링 연동을 지원하는 라이브러리-->
    <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis-spring</artifactId>
        <version>3.0.4</version>
    </dependency>
```

- 2. 스프링 설정파일 작성하기
  - Connection Pool 객체를 스프링 빈으로 등록시킨다.
  - mybatis의 핵심객체(SqlSessionFactory)를 스프링 빈으로 등록시킨다.
  - 매퍼 인터페이스를 전부 스캔해서 매퍼 인스턴스를 생성하고 스프링 빈으로 등록시키는
    MapperScannerConfigurer를 스프링 빈으로 등록시킨다.

많이 나오는 오류

- **Invalid bound statement**
  - **바인드 변수 불일치 많이함**
- **Thers is no getter for property named**
  - **#에 적은게 잘못된거임**

```
    select
            job_id as id,
            job_title as title
            min_salary as minSalary,
            max_salary as maxSalary
```

```
  from jobs
  order by job_id asc

변수명 맞춰주기
```