KH정보교육원

Mybatis

Mybatis

KH정보교육원

Mybatis란?

데이터의 입력, 조회, 수정, 삭제(CRUD)를 보다 편하게 하기 위해 xml 로 구조화한 Mapper 설정 파일을 통해서 JDBC를 구현한 영속성 프레임워크

기존에 JDBC를 통해 구현했던 상당부분의 코드와 파라미터 설정 및 결과 매핑을 xml 설정을 통해 쉽게 구현할 수 있게 해준다.

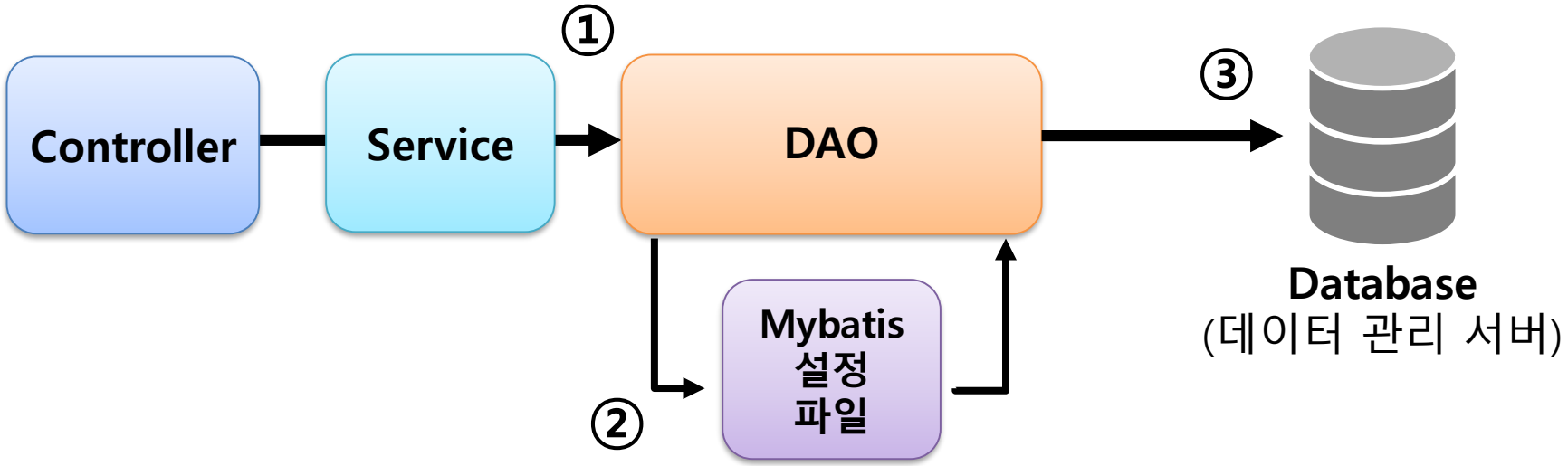
Mybatis API 사이트

http://www.mybatis.org/mybatis-3/ko

KH정보교육원

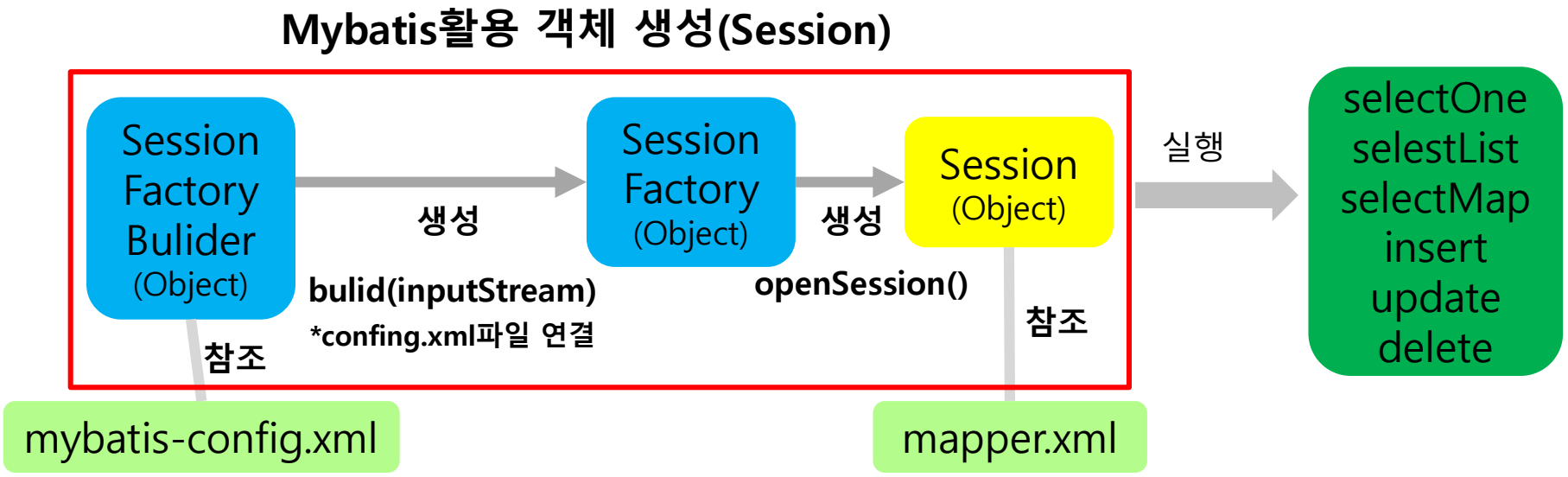
Mybatis의 흐름

이전에 JDBC Template을 통해 SQL을 실행하였다면 Mybatis는 해당 흐름을 전용 라이브러리를 통해 대체하여 동작한다고 생각하면 된다.



Mybatis 동작구조

KH정보교육원



class의 alias 설정
DB연결설정
Sql구문경로 설정(mapper) 등

* 프로젝트에 한 개 존재

SQL 구문설정
인자값, 결과값, 데이터타입 등 설정

* 각 패키지마다 존재
예) 게시판, 멤버 패키지 등

*** 위 객체사용객체는 mybatis-x.x.x.jar 파일에 존재**

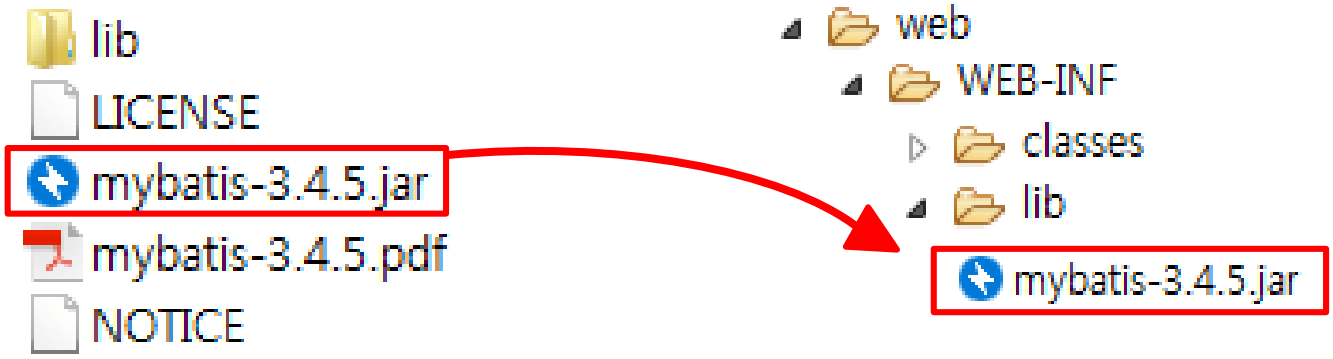
Mybatis 라이브러리 다운 및 연동

아래의 링크에 접속하여 Mybatis 최신 버전을 다운로드 한다.

https://github.com/mybatis/mybatis-3/releases

Mybatis 라이브러리 다운 및 연동

다운로드가 완료되면 압축을 해제하고 mybatis-x.x.x.jar
라이브러리를 프로젝트 내 WEB-INF/lib/ 경로 안에 추가한다.



※ 동봉된 pdf 에 Mybatis의 사용 매뉴얼이 나와 있으니
참고하면 mybatis 사용 시 많은 도움을 받을 수 있다.

Mybatis - 참고

KH정보교육원

ibatis와 Mybatis

기존에 Apache project 에서 ibatis를 운영하던 팀이 2010년 5월 9일에 Google 팀으로 이동하면서 Mybatis로 이름을 바꾸었다.

Mybatis는 기존의 ibatis의 한계점이었던 동적 쿼리와 어노테이션 처리를 보강하여 더 나은 기능을 제공하고 있다.

반대로 ibatis는 현재 비활성화 상태이며, 기존에 ibatis로 만들어진 애플리케이션의 지원을 위해 라이브러리만 제공하고 있다.

Mybatis - 참고

KH정보교육원

ibatis와 Mybatis의 차이점

1. Java 요구 버전

iBatis는 JDK 1.4 이상, MyBatis에서는 JDK 1.5 이상 사용이 가능하다.

2. 패키지 구조 변경

iBatis : com.ibatis.*

MyBatis : org.apache.ibatis.*

3. 사용 용어의 변경

SqlMapConfig	→	Configration
sqlMap	→	Mapper
resultClass	→	resultType

4. 동적 쿼리 지원

Mybatis는 if, choose, trim, foreach 문을 지원한다.

5. 자바 어노테이션 지원

KH정보교육원

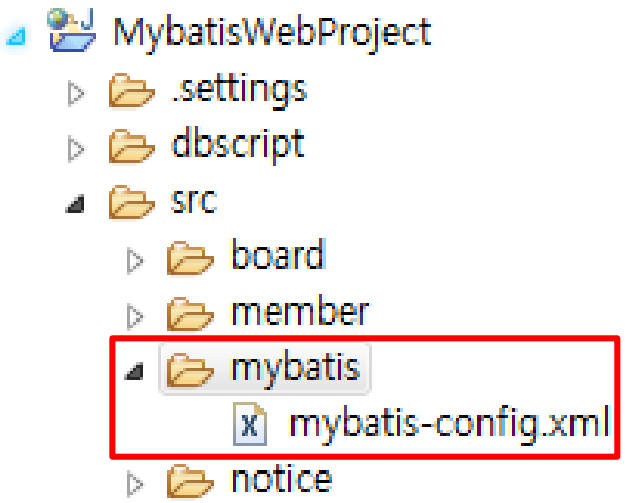
Mybatis-config 설정하기

Mybatis-config 설정하기

KH정보교육원

mybatis-config.xml 생성 위치

src/mybatis 폴더를 생성하고, mybatis-config.xml 파일을 등록한다.




```
<configuration>
  <typeAliases>
    <typeAlias type="mybatissample01.CityInfo" alias="CityInfo"/>
    <typeAlias type="member.model.vo.Member" alias="Member"/>
    <typeAlias type="notice.model.vo.Notice" alias="Notice"/>
    <typeAlias type="board.model.vo.Board" alias="Board"/>
  </typeAliases>
  <environments default="development">
    <environment id="development">
      <transactionManager type="JDBC"/>
      <dataSource type="POOLED">
        <property name="driver" value="oracle.jdbc.driver.OracleDriver"/>
        <property name="url" value="jdbc:oracle:thin:@127.0.0.1:1521:xe"/>
        <property name="username" value="student"/>
        <property name="password" value="student"/>
      </dataSource>
    </environment>
  </environments>
  <mappers>
    <mapper resource="member/model/mapper/member-mapper.xml"/>
  </mappers>
</configuration>
```

Mybatis-config 설정하기

KH정보교육원

mybatis-config.xml 작성

먼저 xml 파일 최상단에 다음과 같이 xml 형식을 지정하여 이하의 설정내용이 mybatis 설정임을 선언한다.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE configuration PUBLIC
“-//mybatis.org//DTD Config 3.0//EN”
“http://mybatis.org/dtd/mybatis-3-config.dtd">
```

<configuration> 최상위 태그를 작성하고 내부에 필요한 설정들을 작성하면 된다.

```
<configuration>

    . . .

</configuration>
```

Mybatis-config 설정하기

KH정보교육원

mybatis-config.xml 작성

<properties> 태그 : 외부 java property파일의 내용을 불러올 때 사용.

<properties> 태그 예시

```
 <properties resource="경로+파일명.properties">
   <!--properties파일에 값 설정 가능-->
   <property name="key명" value="설정값">
</properties>
```

<properties> 설정값 활용

```
<dataSource>
  <property name="명칭" value="${properties에 설정된 key명}" />
  <property name="명칭" value="${properties에 설정된 key명}" />
</dataSource>
```

Mybatis-config 설정하기

KH정보교육원

mybatis-config.xml 작성

<settings> 태그 : mybatis 구동 시 선언할 설정들을 작성한다.

<settings> 태그 예시

```
<settings>
    <!-- Null 값이 발생할 경우 빈칸이 아닌 null로 인식하라 -->
    <setting name="jdbcTypeForNull" value="NULL"/>
</settings>
```

*** 속성값 참조 : http://www.mybatis.org/mybatis-3/ko/configuration.html**

<typeAliases> 태그 : mybatis에서 사용할 자료형의 별칭을 선언한다.

<typeAliases> 태그 예시

```
<typeAliases>
  <!-- type에는 패키지 명까지 전부 기술해주어야 한다. -->
  <typeAlias type="member.model.vo.Member" alias="Member" />
</typeAliases>
```

Mybatis-config 설정하기- 참고

KH정보교육원

Mybatis내장 별칭 (for parameterType / resultType)

Mybatis 타입	Java 자료형	Mybatis 타입	Java 자료형
_byte	byte	double	Double
_long	long	float	Float
_short	short	boolean	Boolean
_int / _integer	int	date	Date
_double	double	object	Object
_float	float	map	Map
_boolean	boolean	hashmap	HashMap
string	String	list	List
byte	Byte	arraylist	ArrayList
long	Long	collection	Collection
short	Short	iterator	Iterator
int / integer	Integer		

Mybatis-config 설정하기

KH정보교육원

mybatis-config.xml 작성

<environments> 태그 : mybatis 에서 연동할 Database 정보를 등록한다.

<environments> 태그 예시

```
<environments default="development">
  <!-- environment id를 구분하여 연결할 DB를 여러 개 구성할 수도 있다 -->
   <environment id="development">
      <transactionManager type="JDBC" />
      <dataSource type="POOLED">
        <property name="driver"
                  value="oracle.jdbc.driver.OracleDriver" />
        <property name="url"
                  value="jdbc:oracle:thin:@127.0.0.1:1521:xe" />
        <property name="username" value="student" />
        <property name="password" value="student" />
      </dataSource>
   </environment>
</environments>
```

*** 여러 개의 DB를 사용할 등록하여 사용할 수 있음**

bulid()메소드 구현시 매개변수에 environment의 id를 설정하면 된다.

Mybatis-config 설정하기 - 참고

KH정보교육원

mybatis-config.xml 작성

POOLED 와 UNPOOLED

Database 연결을 관리하는 DataSource의 타입은 크게 POOLED와 UNPOOLED로 나눌 수 있는데 그 차이는 다음과 같다.

구분	POOLED	UNPOOLED
특징	최초 Connection 객체를 생성 시 해당 정보를 pool 영역에 저장해 두고 이후 Connection 객체를 생성할 때 이를 재 사용한다.	Connection 객체를 별도로 저장하지 않고, 객체 호출 시 매번 생성하여 사용한다.
장점	- Connection 객체를 생성하여 Database와 연결을 구축하는데 걸리는 시간이 단축된다.	- Connection 연결이 많지 않은 코드를 작성할 때 간단하게 구현할 수 있다.
단점	- 단순한 로직을 수행하는 객체를 만들기엔 설정해야 할 정보가 많다.	- 매번 새로운 Connection 객체를 생성하므로 속도가 상대적으로 느리다.

※설정 가능한 type 중 **JNDI**도 있는데, 이는 mybatis에서 Connection 객체를 생성하여 관리하지 않고 Web Application의 설정을 따르겠다는 의미이다.

Mybatis-config 설정하기

KH정보교육원

mybatis-config.xml 작성

<mappers> 태그 : 사용하고자 하는 쿼리가 정의된 mapper 파일을 등록한다.

<mappers> 태그 예시

```
<mappers>
  <mapper resource="member/model/mapper/member-mapper.xml" />
  <mapper resource="notice/model/mapper/notice-mapper.xml" />
  <mapper resource="board/model/mapper/board-mapper.xml" />
</mappers>
```

KH정보교육원

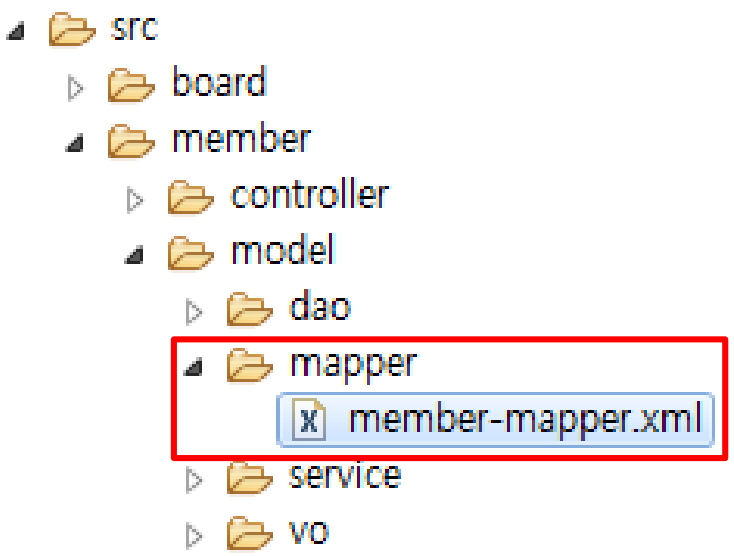
Mapper 설정하기

mapper 설정하기

KH정보교육원

mapper.xml 생성 위치

쿼리 실행이 필요한 model의 위치에 mapper 폴더를 생성하고
식별하기 쉬운 이름을 지어 mapper.xml 파일을 등록한다.




```xml
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE mapper PUBLIC
"-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="mybatis_sample01.CityInfo">

<resultMap id="cityInfoResult" type="CityInfo">
<id property="name" column="Name" />
<result property="code" column="CountryCode" />
</resultMap>

<select id="selectInfo" parameterType="int"
  resultType="CityInfo" resultMap="cityInfoResult">
SELECT * FROM WORLD.CITY WHERE ID = #{ID}
</select>

</mapper>
```

mapper 설정하기

KH정보교육원

mapper.xml 작성

mapper.xml은 사용하고자 하는 쿼리나 결과로 받을 객체를 선언할 수 있다 .

mapper.xml 예시

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE mapper PUBLIC"-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
<mapper namespace="Member">

<resultMap id="resultMember" type="Member">
<id property="id" column="ID" />
<result property="passwd" column="PASSWD" />
</resultMap>

<select id="memberInfo" parameterType="string"
  resultType="_int" resultMap="resultMember">
SELECT * FROM MEMBER WHERE ID = #{userid}
</select>

</mapper>
```

mapper 설정하기

KH정보교육원

mapper.xml 작성

먼저 xml 파일 최상단에 다음과 같이 xml 형식을 지정하여
이하의 설정내용은 mybatis mapper 설정임을 선언한다.

```
<?xml version="1.0" encoding="utf-8"?>
<!DOCTYPE mapper PUBLIC
"-//mybatis.org//DTD Mapper 3.0//EN"
"http://mybatis.org/dtd/mybatis-3-mapper.dtd">
```

이어서 <mapper> 태그를 작성하고, 외부에서 접근할 수 있는 이름인 namespace 속성을 기입한다. 이제 이 후 작성될 태그들은 <mapper> 태그 안에 기록하면 된다.

```
<mapper namespace="Member">

    . . .

</mapper>
```

mapper 설정하기

KH정보교육원

mapper.xml 작성

<resultMap> 태그 : 조회한 결과를 객체와 Row간의 1:1 매칭이 아닌,
원하는 객체의 필드에 담아 반환하고자 할 때 사용한다.

<resultMap> 태그 예시

```
<resultMap id="resultMember" type="Member">
   <!-- prop 는 필드명, column 은 DB 컬럼 명 -->
   <id property="id" column="ID" />
   <result property="passwd" column="PASSWD" />
   . . .
</resultMap>
```

※ resultMap의 type 속성은 실제로 구현해 놓은 자바 POJO 객체를 사용해야 하며, Mybatis-config.xml에서 typeAlias를 지정하지 않은 경우, 패키지 명부터 클래스 명까지 모두 기술해야 한다.

mapper 설정하기

KH정보교육원

mapper.xml 작성

<select> 태그 : SQL의 조회 구문을 작성할 때 쓰인다.
해당 쿼리를 외부에서 접근하고자 할 때 namespace.ID명을 적어 접근이 가능하다.

<select> 태그 예시

```
<select id="memberInfo" parameterType="string“
resultType="_int">

SELECT * FROM MEMBER WHERE ID = #{userid}

<!-- #{field}는 pstmt의 ?의 역할이며, 전달된 값을 뜻한다.
     또한 쿼리의 마지막을 알리는 세미콜론을 찍지 않는다 -->

</select>
```

mapper 설정하기

KH정보교육원

mapper.xml 작성

<select> 태그 주요 속성

속성명	내용
id	구문을 찾기 위해 사용될 수 있는 네임스페이스내 유일한 구분자
parameterType	구문에 전달될 파라미터의 클래스명(패키지 경로 포함)이나 별칭
resultType	리턴되는 타입의 패키지 경로를 포함한 전체 클래스명이나 별칭. collection인 경우 list, arraylist로 설정할 수 있다.
resultMap	사용할 resultMap의 id를 기술한다.

mapper 설정하기

KH정보교육원

mapper.xml 작성

<select> 태그 주요 속성

속성명	내용
flushCache	이 값을 true 로 설정하면 구문이 호출될 때마다 로컬, 2nd 레벨 캐시가 지워진다(flush). 기본값은 false이다.
useCache	이 값을 true 로 설정하면 구문의 결과가 2nd 레벨 캐시에 저장된다. 기본값은 true이다.
timeout	예외가 발생하기 전에 데이터베이스의 요청 결과를 기다리는 최대시간을 설정한다. 드라이버에 따라 다소 지원되지 않을 수 있다.
statementType	STATEMENT, PREPARED 또는 CALLABLE 중 하나를 선택할 수 있다. 마이바티스에게 Statement, PreparedStatement 또는 CallableStatement를 사용하게 한다. 기본값은 PREPARED이다.

※ resultMap과 resultType은 둘 모두를 사용할 수 없으며,
둘 중 하나만 선언해야 한다.

mapper 설정하기 - 참고

KH정보교육원

mapper.xml 작성

FlushCache? useCache?

일반적으로 쿼리를 수행할 때, 쿼리의 결과나 호출되는 내용이 변동이 없는 정적인 쿼리나 결과라면, 이를 매 반복 시마다 굳이 새로운 쿼리로 생성하여 호출하거나, 새로운 결과를 받아 올 필요가 없을 것이다.

이러한 상황을 위해 Mybatis에서는 Cache 라는 저장소를 내장하여, 반복되는 정적인 쿼리의 호출이나 결과에 대한 내용을 한 번 이상 실행할 경우 이를 미리 저장해두어 재 호출에 소요되는 시간을 절약할 수 있게 도와준다.

mapper 설정하기

KH정보교육원

mapper.xml 작성

<insert>, <update>, <delete> 태그

: 해당 태그들은 설정이 동일하다.

<insert> 태그 예시

```
<insert id="insertMember" parameterType="Member"
flushCache="true"
 statementType="PREPARED" keyProperty="" keyColumn=""
useGeneratedKeys="true" timeout="20">
   INSERT INTO MEMBER VALUES(
      #{id}, #{passwd}, #{name},
      #{email}, #{gender}, #{age},
      #{phone}, #{address}, DEFAULT
   )
   </insert>
```

객체를 파라미터를 받는 경우 해당 객체의 필드 변수를 '변수명 = 값'의 Map 방식으로 조회하여 가져올 수 있다.

mapper 설정하기

KH정보교육원

mapper.xml 작성

<update> 태그 예시

```
<update id="updateMember" parameterType="Member"
flushCache="true" statementType="PREPARED" timeout="20">
  UPDATE MEMBER
  SET PASSWD = #{passwd}, EMAIL = #{email}, AGE = #{age},
  PHONE = #{phone}, ADDRESS = #{address}
  WHERE ID = #{id}
</update>
```

<delete> 태그 예시

```
<delete id="deleteMember" parameterType="string"
flushCache="true" statementType="PREPARED" timeout="20">
  DELETE FROM MEMBER WHERE ID = #{userid}
</delete>
```

mapper 설정하기

KH정보교육원

mapper.xml 작성

<insert>, <update>, <delete> 태그 주요 속성

id	구문을 찾기 위해 사용될 수 있는 네임스페이스내 유일한 구분자
parameterType	구문에 전달될 파라미터의 클래스명(패키지 경로 포함)이나 별칭
flushCache	이 값을 true 로 셋팅하면 구문이 호출될때마다 캐시가 지원질것이다(flush). 디폴트는 false 이다.
timeout	예외가 발생하기 전에 데이터베이스의 요청 결과를 기다리는 최대시간을 설정한다. 드라이버에 따라 다소 지원되지 않을 수 있다.
useGeneratedKeys	(insert, update 에만 적용) 데이터베이스에서 내부적으로 생성한 키(예를 들어 MySQL또는 SQL Server의 자동 증가 필드)를 받는 JDBC getGeneratedKeys 메소드를 사용하도록 설정한다. 기본값은 false.
keyProperty	(insert, update 에만 적용) getGeneratedKeys 메소드나 insert 구문의 selectKey태그의 설정 select문의 결과를 저장할 프로퍼티를 지정. 디폴트는 셋팅하지 않는 것이다. 여러 개의 컬럼을 사용한다면 프로퍼티명에 콤마를 구분자로 나열할 수 있다.

KH정보교육원

Mybatis 활용하기

Mybatis 활용하기

KH정보교육원

Mybatis SqlSessionFactory 생성

Mybatis-config.xml, mapper.xml 파일 생성을 완료했다면, DAO에서 세션 생성을 위한 getSqlSessionFactory() 메소드를 작성한다.

```
private SqlSessionFactory getSqlSessionFactory() {

	String resource = "mybatis/mybatis-config.xml";
	SqlSessionFactory factory = null;
	try {
		InputStream inputStream =
			Resources.getResourceAsStream(resource);
		SqlSessionFactoryBuilder builder =
			new SqlSessionFactoryBuilder();
		factory = builder.build(inputStream);
	} catch (IOException e) {
	}
	return factory;
}
```

mybatis-config.xml 의 설정 정보를 InputStream 객체를 통해 읽어와 SqlSessionFactory 객체를 생성한다.

Mybatis 활용하기

KH정보교육원

SqlSessionFactoryBuilder 매소드

메소드 구분	설명
build(InputStream)	config.xml파일만 불러옴
build(InputStream, String)	config.xml파일과 지정한 DB를 불러옴
build(InputStream, Properties)	config.xml파일과 프로퍼티로 설정한 내용으로 불러옴("${key명}")
build(InputStream, String, Properties)	config.xml파일과 지정한 DB, properties파일 불러옴
build(configuration)	configurateion객체에 설정한 내용을 불러옴

* config.xml은 Resource객체의 getResourceAsStream매소드를 이용 InputStream으로 가져옴

Mybatis 활용하기

KH정보교육원

Mybatis SqlSession 생성

DAO 메소드는 SqlSqssionFactory를 사용하여 SqlSession 객체를 생성한다.

```
public Member selectMember(Member m) {
    Member member = null;
    SqlSession session = null;

    try {
        session = getSqlSessionFactory().openSession(false);

        // 매퍼 파일 안에 작성된 쿼리문 실행시키고 결과를 받는다.
        member = session.selectOne("Member.loginCheck", m);
        System.out.println(session);
    } catch (Exception e) {
        e.printStackTrace();
    } finally{
        session.close();
    }
    return member;
}
```

getSqlSessionFactory() 메소드를 통해
세션 정보를 받아와 SqlSession을 생성한다.

Mybatis SqlSession

Mybatis SqlSessionFactory를 통해 세션을 생성하는 openSession() 메소드는 여러가지 형식으로 오버로딩 되어 있는데, 그 중 대표적인 것은 다음과 같다.

메소드 구분	설명
openSession()	기본값을 통해 SqlSession을 생성한다.
openSession(Boolean)	SqlSession 생성 시 AutoCommit 여부를 true / false 로 지정할 수 있다. (기본값은 true)
openSession(Connection)	직접 생성한 Connection 객체를 이용해 SqlSession을 생성한다. (기본값 : X)
openSession(ExecutorType)	쿼리를 실행할 때 PreparedStatement의 재사용 여부를 설정한다. (기본값 : ExecutorType.SIMPLE)

KH정보교육원

SqlSession을 통한 쿼리 실행 예시

생성한 SqlSession 객체의 메소드를 통해 정의한 쿼리에 접근한다.

```
public Member selectMember(Member m) {
    Member member = null;
    SqlSession session = null;

    try {
        session = getSqlSessionFactory().openSession(false);

        // 매퍼 파일 안에 작성된 쿼리문 실행시키고 결과를 받는다.
        member = session.selectOne("Member.loginCheck", m);
        System.out.println(session);
    } catch (Exception e) {
        e.printStackTrace();
    } finally{
        session.close();
    }
    return member;
}
```

mapper에 정의된 namespace명.query_ID
를 통해 정의한 select문에 접근한다.

Mybatis 활용하기

KH정보교육원

SqlSession을 통한 mapper 쿼리 실행

mapper.xml에서 선언한 쿼리구문을 SqlSession에서 실행하는 메소드는 총 6가지가 있다.

메소드 명	설명
`Object selectOne (String mapper, Object param)`	하나의 객체만을 받고자 할 때 사용
`List<E> selectList (String mapper, Object param)`	결과에 대한 값을 List로 받고자 할 때 사용
`Map<K,V> selectMap (String mapper, Object param, String mapKey)`	결과에 대한 값을 Map으로 받고자 할 때 사용, (마지막 인자로 키로 사용될 컬럼을 명시한다)
`int insert (String mapper, Object param)`	Database에 데이터를 입력하고자 할 때 사용
`int update (String mapper, Object param)`	Database의 데이터를 수정하고자 할 때 사용
`int delete (String mapper, Object param)`	Database의 데이터를 삭제하고자 할 때 사용