



# **MyBatis**

# 목차

- ✓ Chap01. MyBatis 개념 및 흐름
- ✓ Chap02. mybatis-config.xml 설정하기
- ✓ Chap03. mapper 설정하기
- ✓ Chap04. MyBatis 활용하기

# MyBatis의 개념 및 흐름

# ▶ MyBatis

## ✓ MyBatis 란 ??

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

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

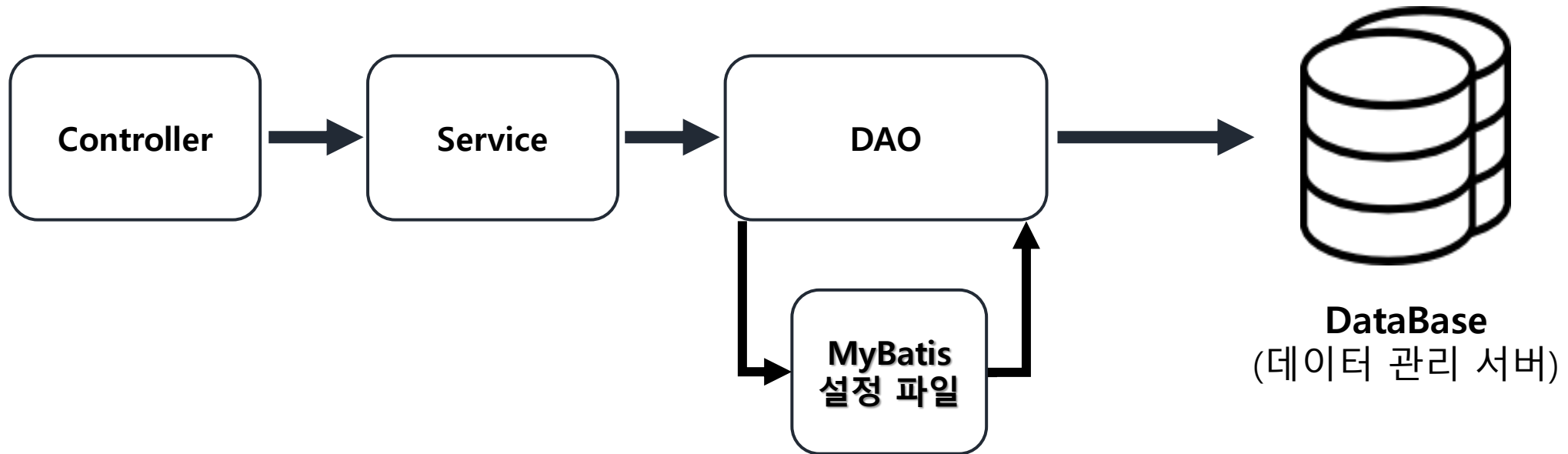
## ✓ MyBatis API 사이트

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

# ▶ MyBatis

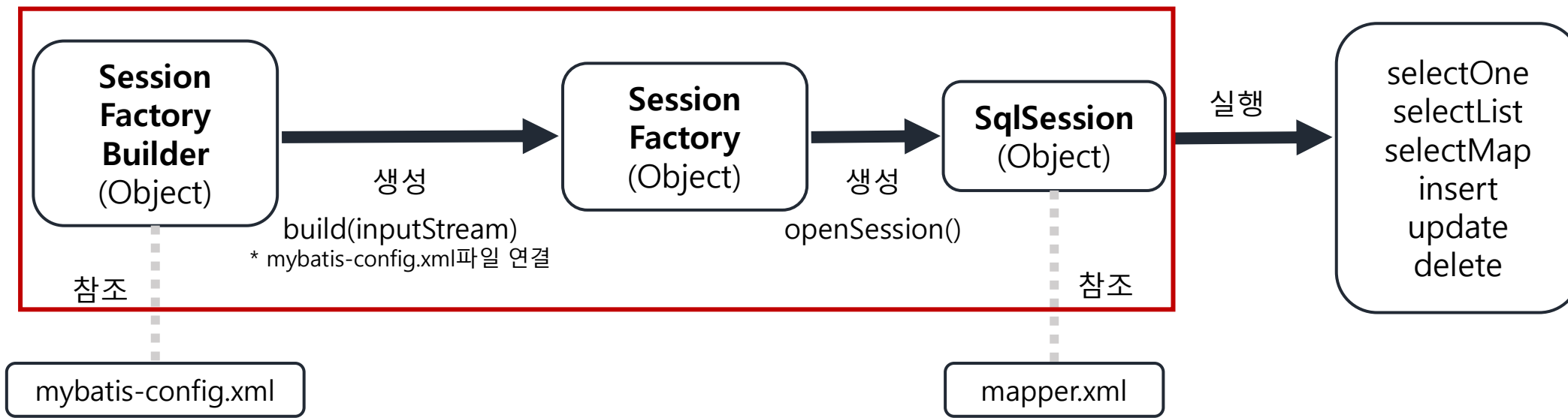
## ✓ MyBatis의 흐름

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



# ▶ MyBatis의 동작 구조

## MyBatis 활용 객체 생성 (Session)



- Class의 Alias(별칭) 설정
- DB 연결 설정
- Sql 구문 경로 설정

- SQL 쿼리문 설정  
(인자값, 결과값, 데이터타입 등 설정)

- \* 각 패키지 마다 존재  
예) 게시판, 멤버 패키지 등등

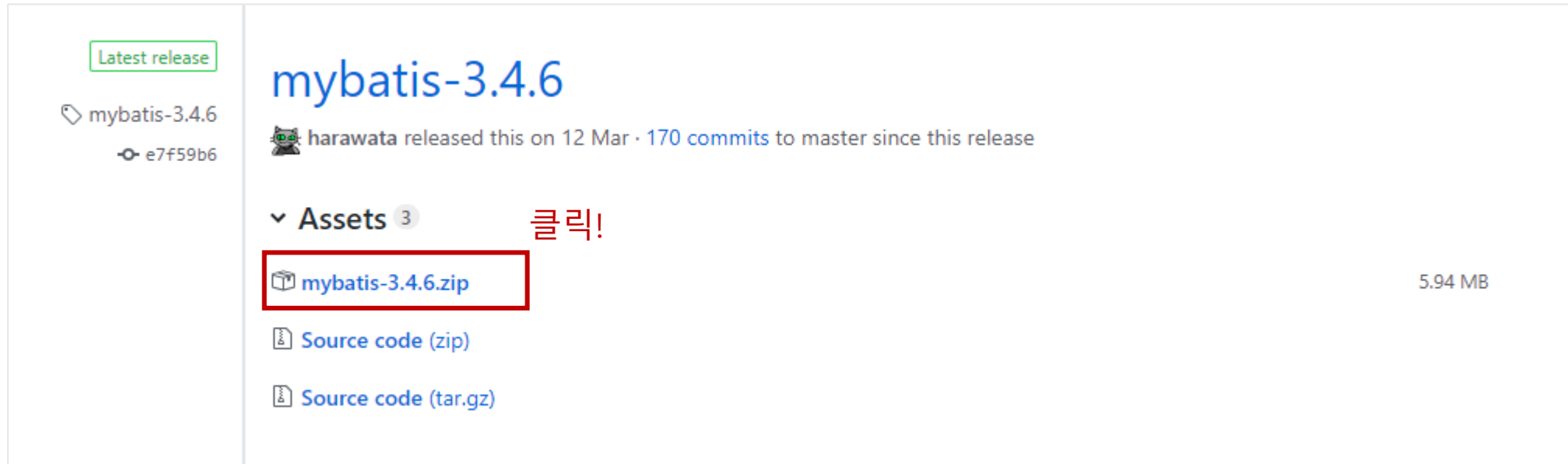
\* 위의 사용 객체들은 mybatis-x.x.x.jar 파일에 존재

# ▶ MyBatis 라이브러리

## ✓ MyBatis 라이브러리 다운 및 연동

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

위의 링크에 접속하여 mybatis-.x.x.x 버전을 다운로드



Latest release

mybatis-3.4.6  
e7f59b6

mybatis-3.4.6

harawata released this on 12 Mar · 170 commits to master since this release

Assets 3

mybatis-3.4.6.zip 5.94 MB

Source code (zip)

Source code (tar.gz)

클릭!

# ▶ MyBatis 라이브러리

## ✓ MyBatis 라이브러리 다운 및 연동

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





## ▶ MyBatis - 참고

### ✓ ibatis와 MyBatis

기존에 Apache project에서 ibatis를 운영하던 팀이 2010년 5월 9일에 Google 팀으로 이동하면서 MyBatis로 이름을 바꿈

MyBatis는 기존의 ibatis의 한계점이었던 동적 쿼리와 어노테이션 처리를 보강하여 더 나은 기능을 제공

반대로 ibatis는 현재 비활성화 상태이며, 기존에 ibatis로 만들어진 애플리케이션의 지원을 위해 라이브러리만을 제공하고 있음

# ▶ MyBatis - 참고

## ✓ 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	➡	Configuration
sqlMap	➡	Mapper
resultClass	➡	resultType

### 4. 동적 쿼리 지원

MyBatis는 if, choose, trim, foreach 문을 지원한다.

### 5. 자바 어노테이션 지원

# mybatis-config 설정하기

# ▶ mybatis-config 설정하기

## ✓ mybatis-config.xml 생성 위치

'resources'라는 Source Folder를 생성하고, mybatis-config.xml 파일 등록



```
<?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>
  <environments default="development">
    <environment id="development">
      <transactionManager type="JDBC"/>
      <dataSource type="POOLED">
        <property name="driver" value="${driver}"/>
        <property name="url" value="${url}"/>
        <property name="username" value="${username}"/>
        <property name="password" value="${password}"/>
      </dataSource>
    </environment>
  </environments>
  <mappers>
    <mapper resource="org/mybatis/example/BlogMapper.xml"/>
  </mappers>
</configuration>
```

# ▶ mybatis-config 설정하기

## ✓ mybatis-config.xml 작성

- xml 최상단에 다음과 같이 xml 형식을 지정하여 이하의 설정 내용이 mybatis config 설정임을 선언

```
<?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 됨

```
<configuration>
```

```
  <!-- 내부에 필요한 설정들 작성 -->
```

```
</configuration>
```

# ▶ mybatis-config 설정하기

## ✓ mybatis-config.xml 작성

\* <properties> 태그 : 외부 properties 파일의 내용을 불러올 때 사용하는 태그

- <properties> 태그 예시

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

- <properties> 설정 값 활용

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

# ▶ mybatis-config 설정하기

## ✓ mybatis-config.xml 작성

\* **<settings>** 태그 : mybatis 구동 시 선언할 설정들을 작성하는 태그

- <settings> 태그 예시

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

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

\* **<typeAliases>** 태그 : mybatis에서 사용할 자료형의 별칭을 선언하는 태그

- <typeAliases> 태그 예시

```
<typeAliases>
  <typeAlias type="com.br.mybatis.member.model.vo.Member" alias="Member" />
  <typeAlias type="com.br.mybatis.board.model.vo.Board" alias="Board" />
  <typeAlias type="com.br.mybatis.board.model.vo.Reply" alias="Reply" />
</typeAliases>
```

# ▶ mybatis-config 설정하기 - 참고

## ✓ MyBatis 내장 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

## ✓ mybatis-config.xml 작성

\* <environments> 태그 : mybatis에서 연동할 DataBase 정보를 등록하는 태그

- <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:@localhost:1521:xe"/>
      <property name="username" value="mybatis"/>
      <property name="password" value="mybatis"/>
    </dataSource>
  </environment>
</environments>
```

\* 여러 개의 DB를 등록하여 사용할 수 있음

- build() 메소드 구현 시 매개변수에 environment의 id를 설정하면 됨

# ▶ mybatis-config 설정하기 - 참고

## ✓ POOLED와 UNPOOLED의 차이점

구분	POOLED	UNPOOLED
특징	최초 Connection 객체를 생성할 때 해당 정보를 pool 영역에 저장해두고 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 구현 할 수 있다.
단점	단순한 로직을 수행하는 객체를 만들기에는 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

## ✓ mybatis-config.xml 작성

\* <mappers> 태그 : 사용하고자 하는 쿼리문이 정의된 mapper 파일을 등록하는 태그

- <mappers> 태그 예시

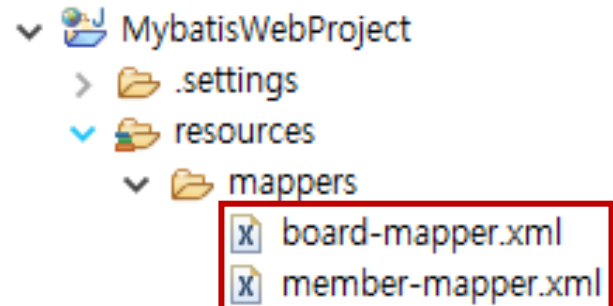
```
<mappers>
  <mapper resource="resources/mappers/member-mapper.xml"/>
  <mapper resource="resources/mappers/board-mapper.xml"/>
</mappers>
```

# mapper 설정하기

# ▶ mapper 설정하기

## ✓ \*-mapper.xml 생성 위치

'resources' 폴더 안에 'mappers' 폴더를 생성하고 그 안에 식별하기 쉬운 이름을 지어 파일을 등록



# ▶ mapper 설정하기

## ✓ \*-mapper.xml 작성

- xml 최상단에 다음과 같이 xml 형식을 지정하여 이하의 설정 내용이 mybatis mapper 설정임을 선언

```
<?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 기입, 이 후 작성될 태그들은 <mapper>태그 안에 기록하면 됨

```
<mapper namespace="memberMapper">

    <!-- mapper 내부에 작성될 내용 -->

</mapper>
```

# ▶ mapper 설정하기

## ✓ \*-mapper.xml 작성

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

- <resultMap> 태그 예시

```
<resultMap type="Member" id="memberResultSet">
  <!-- property = 자바의 필드변수 이름 / column = DB의 해당 컬럼 -->
  <!-- id는 primary key / result는 일반 컬럼 -->
  <id property="mid" column="MID"/>
  <result property="userId" column="USER_ID"/>
  <result property="userPwd" column="USER_PWD"/>
  <result property="userName" column="USER_NAME"/>
</resultMap>
```

\* <resultMap>의 type 속성은 실제로 구현해 놓은 자바 POJO 객체를 사용해야 하며, mybatis-config.xml에서 typeAlias를 지정하지 않은 경우, 패키지 명부터 클래스 명 까지 모두 기술해야 됨

# ▶ mapper 설정하기

## ✓ \*-mapper.xml 작성

\* <select> 태그 : SQL의 조회 구문을 작성할 때 사용되는 태그로, 해당 쿼리를 외부에서 접근하고자 할 때 namespace.id 명을 적어 접근 가능

- <select> 태그 예시

```
<select id="memberInfo" parameterType="string" resultType="_int">
    <!-- #{field}는 pstmt의 '?'의 역할이며, 전달된 값을 뜻함
        또한 여러줄로 줄바꿈문자를 섞어 사용도 가능하다.
        단, 쿼리의 마지막을 알리는 세미콜론은 예러를 유발한다. -->
    SELECT *
    FROM MEMBER
    WHERE USER_ID = #{userId}
</select>
```



## ▶ mapper 설정하기 - 참고

### ✓ <select> 태그 주요 속성

속성명	내용
<b>id</b>	구문을 찾기 위해 사용될 수 있는 네임스페이스 내 유일한 구분자
<b>parameterType</b>	구문에 전달될 파라미터의 클래스 명(패키지 경로 포함)이나 별칭
<b>resultType</b>	리턴되는 타입의 패키지 경로를 포함한 전체 클래스 명이나 별칭, collection인 경우 list, arraylist로 설정할 수 있다.
<b>resultMap</b>	사용할 resultMap의 id를 기술한다.

\* resultMap과 resultType은 둘 모두를 사용할 수 없으며, 둘 중 하나만 선언해야 된다.

## ▶ mapper 설정하기 - 참고

### ✓ <select> 태그 주요 속성

속성명	내용
<b>flushCache</b>	이 값을 true로 설정하면 구문이 호출 될 때마다 로컬, 2 <sup>nd</sup> 레벨 캐시가 지워진다(flush). (기본 값 : false)
<b>useCache</b>	이 값을 true로 설정하면 구문의 결과가 2 <sup>nd</sup> 레벨 캐시에 저장된다. (기본 값 : true)
<b>timeout</b>	예외가 발생하기 전에 데이터베이스의 요청 결과를 기다리는 최대 시간을 설정한다. 드라이버에 따라 다소 지원되지 않을 수 있다.
<b>statementType</b>	STATEMENT, PREPARED 또는 CALLABLE 중 하나를 선택할 수 있다. MyBatis에게 Statement, PreparedStatement 또는 CallableStatement를 사용하게 한다. (기본 값 : PREPARED)

## ▶ mapper 설정하기 - 참고

### ✓ <insert>, <update>, <delete> 태그 주요 속성

속성명	내용
id	구문을 찾기 위해 사용될 수 있는 네임스페이스 내 유일한 구분자
parameterType	구문에 전달될 파라미터의 클래스 명(패키지 경로 포함)이나 별칭
flushCache	이 값을 true로 설정하면 구문이 호출 될 때마다 캐시가 지워진다(flush). (기본 값 : false)
timeout	예외가 발생하기 전에 데이터베이스의 요청 결과를 기다리는 최대 시간을 설정한다. 드라이버에 따라 다소 지원되지 않을 수 있다.
userGeneratedKeys	(insert, update에만 적용) 데이터베이스에서 내부적으로 생성한 키(예를 들어 MySQL 또는 SQL Server의 자동 증가 필드)를 받는 JDBC getGeneratedKeys 메소드를 사용하도록 설정한다. (기본 값 : false)
keyProperty	(insert, update에만 적용) getGeneratedKeys 메소드나 insert 구문의 selectKey 태그의 설정 select 문의 결과를 저장할 프로퍼티를 지정. 디폴트는 셋팅하지 않는 것이다. 여러 개의 컬럼을 사용한다면 프로퍼티명에 콤마를 구분자로 나열할 수 있다.

# MyBatis 활용하기

# ▶ SqlSession 생성하기

## ✓ 싱글톤을 적용한 Template 클래스 생성

mybatis-config.xml, \*-mapper.xml 파일 생성을 완료했다면, common 패키지를 만들어 싱글톤을 적용한 Template 클래스를 만들고 SqlSession을 반환해주는 static 메소드를 작성한다.

```
public class Template {  
  
    public static SqlSession getSqlSession() {  
        SqlSession session = null;  
  
        String resource = "/mybatis-config.xml";  
  
        try {  
            InputStream stream = Resources.getResourceAsStream(resource);  
  
            SqlSessionFactoryBuilder builder = new SqlSessionFactoryBuilder();  
            SqlSessionFactory factory = builder.build(stream);  
  
            session = factory.openSession(false);  
  
        } catch (IOException e) {  
            e.printStackTrace();  
        }  
  
        return session;  
    }  
}
```

1. mybatis-config.xml의 설정 정보를 InputStream 객체를 통해 읽어옴
2. SqlSessionFactoryBuilder 객체를 생성하고 build() 메소드를 통해 SqlSessionFactory 객체를 생성
3. SqlSessionFactory 객체의 openSession() 메소드를 통해 SqlSession 객체 생성

## ▶ SqlSession 생성하기 - 참고

### ✓ SqlSessionFactoryBuilder 메소드

메소드	설명
<b>build(InputStream)</b>	config.xml 파일만 불러옴
<b>build(InputStream, String)</b>	config.xml 파일과 지정한 DB를 불러옴
<b>build(InputStream, Properties)</b>	config.xml 파일과 프로퍼티로 설정한 내용으로 불러옴 ("\${ key명 }")
<b>build(InputStream, String, Properties)</b>	config.xml 파일과 지정한 DB, Properties 파일을 불러옴
<b>build(configuration)</b>	configuration 객체에 설정한 내용을 불러옴

\* config.xml은 Resource 객체의 getResourceAsStream 메소드를 이용하여 InputStream으로 가져옴

# ▶ SqlSession 생성하기 - 참고

## ✓ SqlSessionFactory 메소드

메소드	설명
<b>openSession()</b>	기본값을 통해 SqlSession을 생성한다.
<b>openSession(Boolean)</b>	SqlSession 생성 시 AutoCommit 여부를 true / false로 지정할 수 있다. (기본 값 : true)
<b>openSession(Connection)</b>	직접 생성한 Connection 객체를 이용해 SqlSession을 생성한다. (기본 값 : X)
<b>openSession(ExecutorType)</b>	쿼리를 실행 할 때 PreparedStatement의 재사용 여부를 설정한다. (기본 값 : ExecutorType.SIMPLE)

# ▶ MyBatis 활용하기

## ✓ SqlSession을 통한 쿼리 실행

1. Service 클래스에서 getSession 메소드 호출을 통해 SqlSession 생성
2. Dao 클래스의 메소드 호출 시 전달 인자로 SqlSession 객체 전달
3. Dao 클래스의 메소드에서 SqlSession 객체를 통해 쿼리에 접근

```
public Member selectMember(SqlSession session, Member m) throws LoginFailException {  
    // 리턴용 멤버 객체 선언  
    Member member = null;  
  
    member = session.selectOne("memberMapper.loginMember", m);  
  
    if (member == null) {  
        * mapper에 정의된 namespace명. query_id명을 통해 정의한 select문에 접근  
        session.close();  
        throw new LoginFailException("로그인 실패!!");  
    }  
  
    return member;  
}
```



# ▶ MyBatis 활용하기 - 참고

## ✓ SqlSession을 통한 쿼리 실행

메소드	반환형	설명
<code>selectOne(String mapper, Object param)</code>	<code>Object</code>	하나의 객체만을 받고자 할 때 사용
<code>selectList(String mapper, Object param)</code>	<code>List&lt;E&gt;</code>	결과에 대한 값을 List로 받고자 할 때 사용
<code>selectMap(String mapper, Object param, String mapKey)</code>	<code>Map&lt;K,V&gt;</code>	결과에 대한 값을 Map으로 받고자 할 때 사용 (마지막 인자로 키로 사용될 컬럼을 명시)
<code>insert(String mapper, Object param)</code>	<code>int</code>	DB에 데이터를 입력하고자 할 때 사용
<code>update(String mapper, Object param)</code>	<code>int</code>	DB의 데이터를 수정하고자 할 때 사용
<code>delete(String mapper, Object param)</code>	<code>int</code>	DB의 데이터를 삭제하고자 할 때 사용