Mybatis

목차

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.

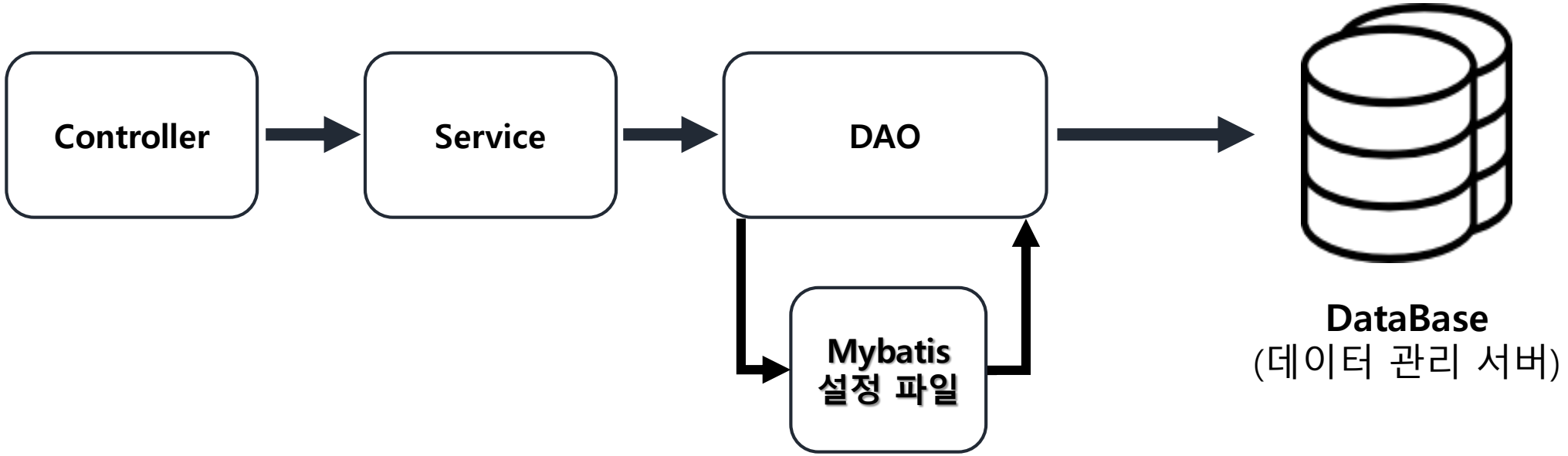
✓ Mybatis API 사이트

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.



▶ JDBC, Spring, Mybatis 관련 모듈,라이브러리 추가

JDBC 드라이버 (ojdbc11)

https://mvnrepository.com/artifact/com.oracle.database.jdbc/ojdbc11

Spring에서 JDBC지원하는 모듈 (spring-jdbc)

https://mvnrepository.com/artifact/org.springframework/spring-jdbc

Mybatis 라이브러리 (mybatis)

https://mvnrepository.com/artifact/org.mybatis/mybatis

Spring에서 쉽게 Mybatis를 사용 가능하게 만드는 모듈(spring-mybatis)

https://mvnrepository.com/artifact/org.mybatis/mybatis-spring

DataBase Connection Pool 사용을 위한 라이브러리(commons-dbcp2)

https://mvnrepository.com/artifact/org.apache.commons/commons-dbcp2

▶ Mybatis - 참고

✓ ibatis와 Mybatis

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

Mybatis는 기존의 ibatis의 한계점이였던 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	→	Configration
sqlMap	→	Mapper
resultClass	→	resultType

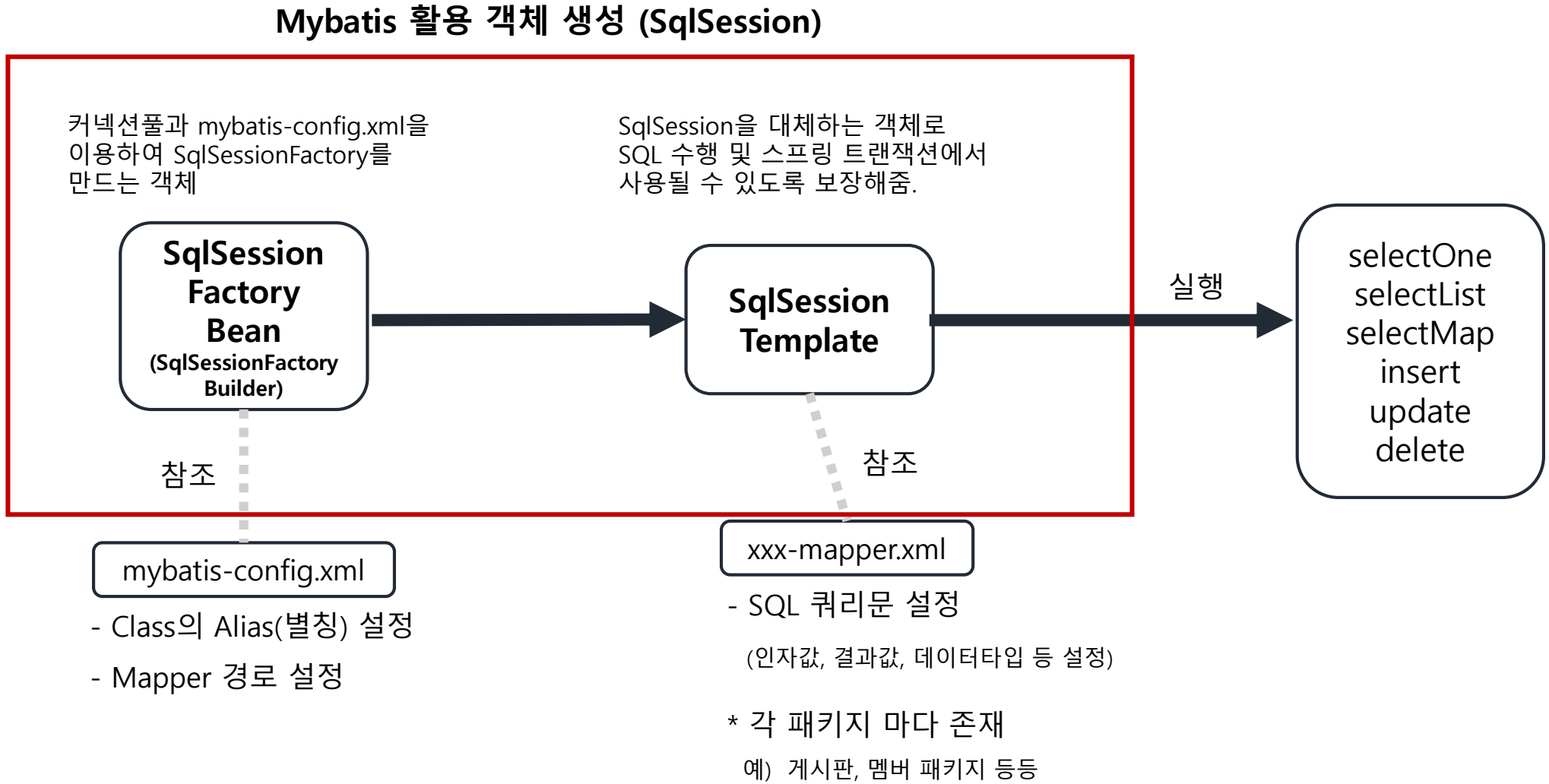
4. 동적 쿼리 지원

Mybatis는 if, choose, trim, foreach 문을 지원한다.

5. 자바 어노테이션 지원

mybatis-config 설정하기

▶ Mybatis의 동작 구조



*** 위의 사용 객체들은 mybatis 관련 라이브러리에 존재함**

▶ mybatis-config 설정하기

✓ Mybatis 설정용 xml 파일 DTD(Document Type Definition) 추가

Window - Preferences - XML - XML Catalog

- User Specified Entries클릭 – Add... 클릭

- Config

Location : http://mybatis.org/dtd/mybatis-3-config.dtd

Key type : Public ID

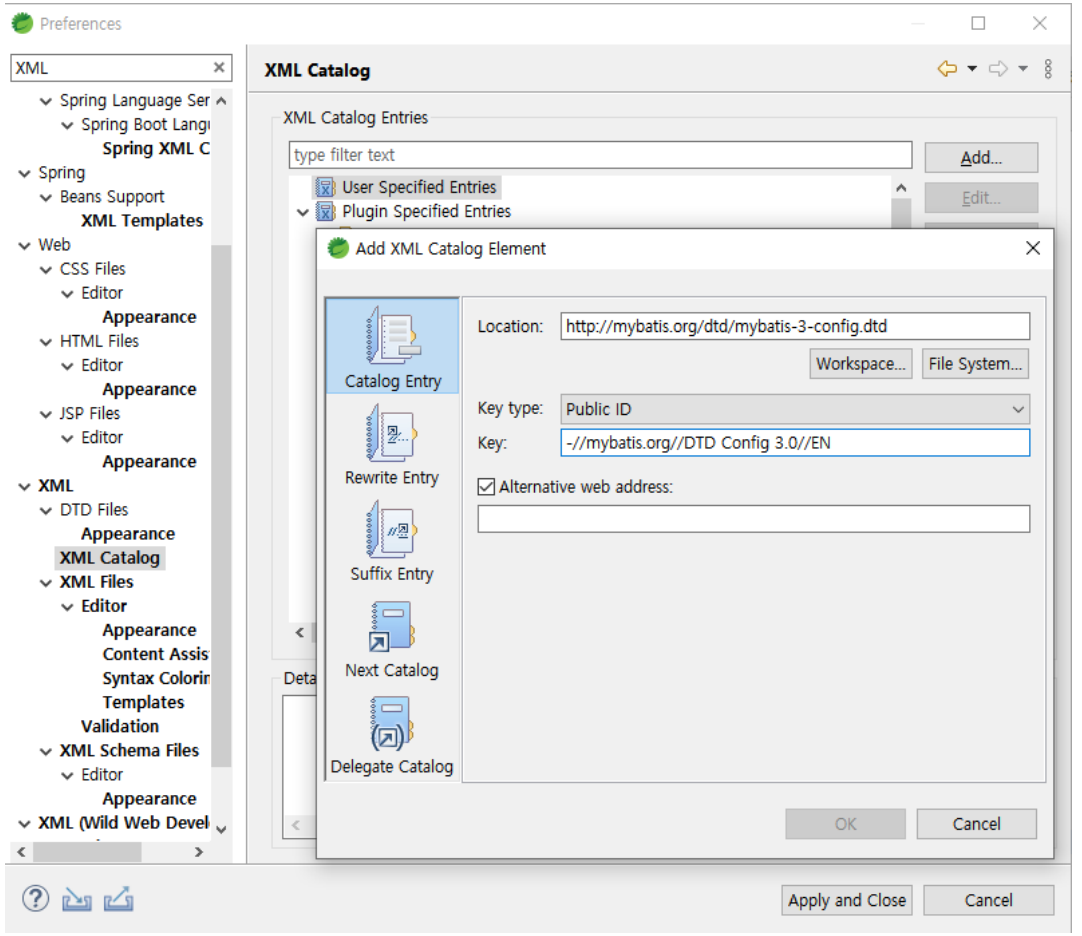
Key : -//mybatis.org//DTD Config 3.0//EN

- Mapper

Location : http://mybatis.org/dtd/mybatis-3-mapper.dtd

Key type : Public ID

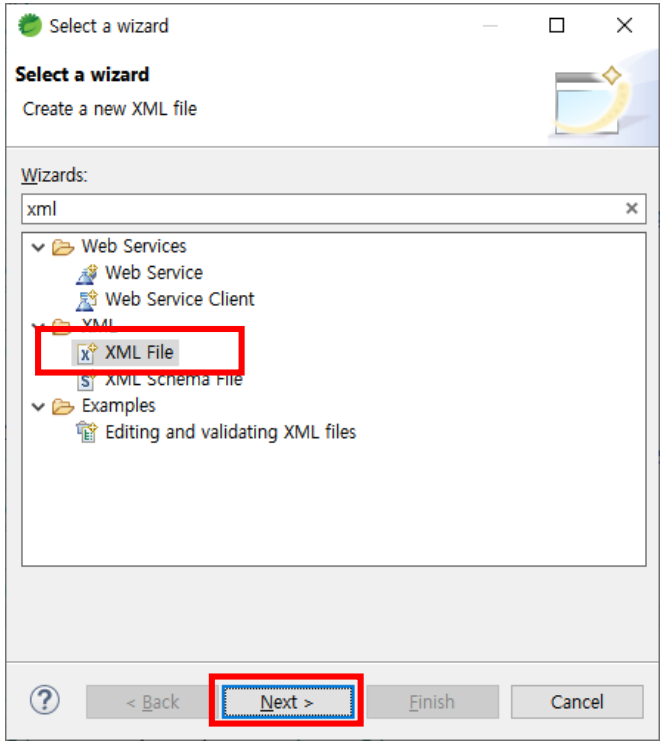
Key : -//mybatis.org//DTD Mapper 3.0//EN

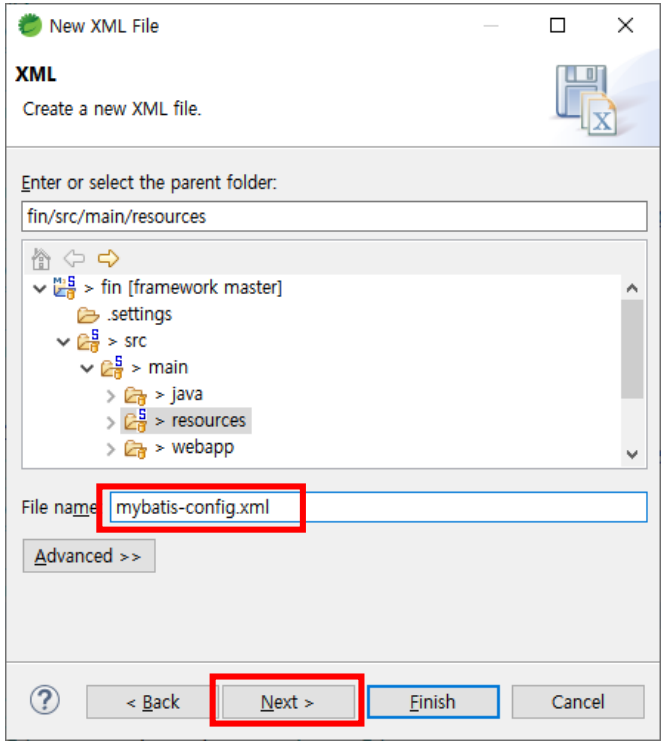


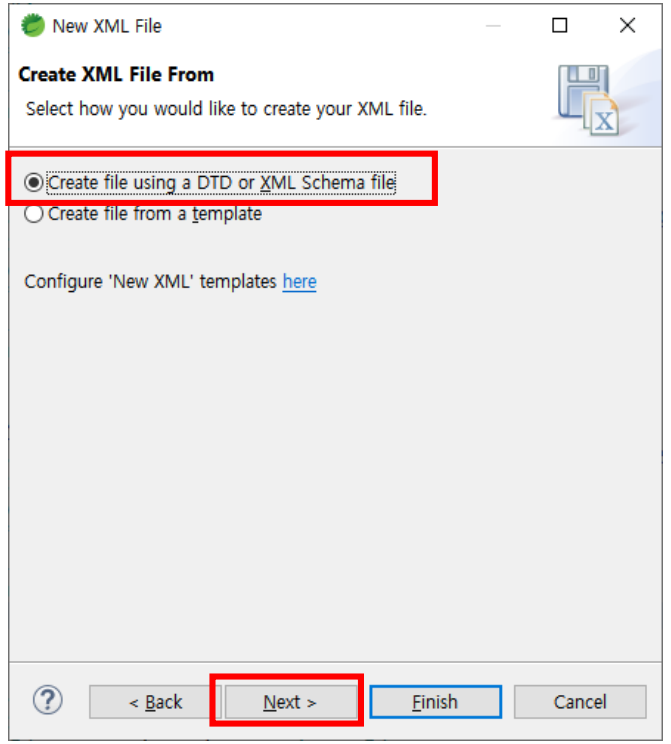
▶ mybatis-config 설정하기

✓ mybatis-config.xml 생성

'src/main/resources'폴더에 mybatis-config.xml 파일 생성



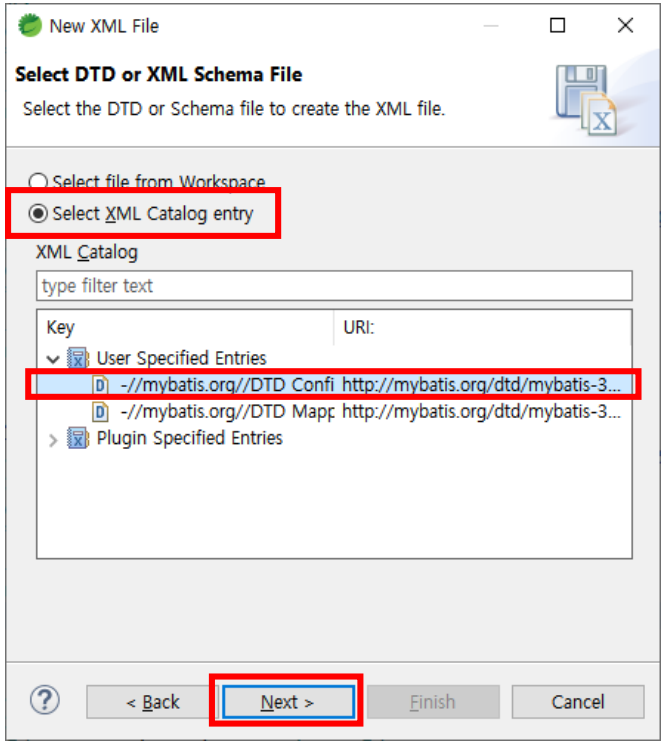


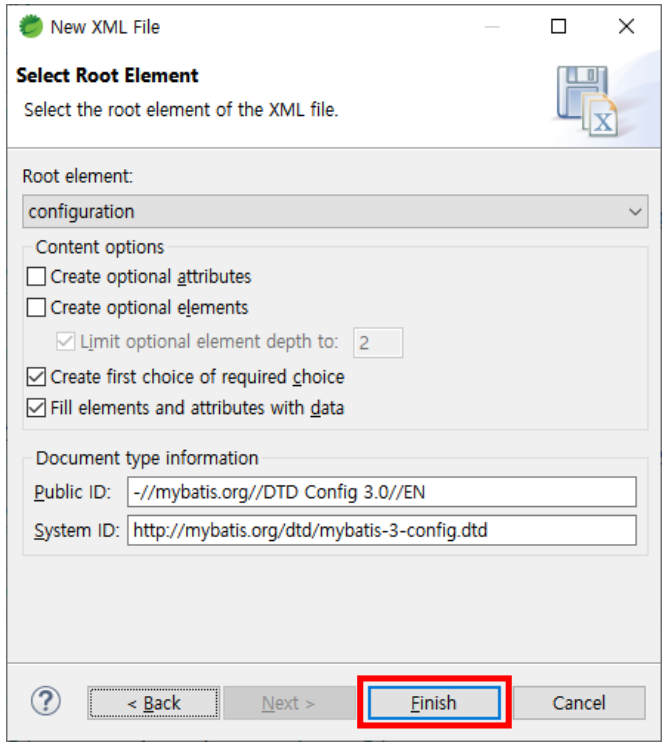


▶ mybatis-config 설정하기

✓ mybatis-config.xml 생성

'src/main/resources'폴더에 mybatis-config.xml 파일 생성





▶ mybatis-config 설정하기

✓ mybatis-config.xml 작성

```xml
<?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>

	<!-- SqlSessionTemplate 생성 시 적용될 설정 작성 부분 -->
	<settings>
		<!-- insert 또는 update에 사용되는 값 중 null이 있을 경우에 대한 설정
			해당 설정이 없을 경우 -> SQL 구문에 null 포함되어 있다는 오류 발생.
			해상 설정이 있을 경우 -> 오류를 발생 시키지 않고 NULL 값을 컬럼에 대입
						단, NOT NULL 제약조건이 없는 컬럼에만 가능함.

			** value 설정 시 NULL 은 반드시 대문자로 작성 (소문자 null은 오류가 발생함) -->
		<setting name="jdbcTypeForNull" value="NULL"/>
	</settings>

	<!-- 별칭 작성 부분  -->
	<!-- VO클래스의 패키지명 + 클래스명 작성하는 것이 불편하기 때문에 짧은 별칭 부여 -->
	<typeAliases>
		<typeAlias type="edu.kh.fin.member.model.vo.Member" alias="Member"/>
	</typeAliases>

	<!-- mapper 파일(SQL 작성되는파일) 위치 등록 부분 -->
	<mappers>
		<mapper resource="/mappers/member-mapper.xml"/>
	</mappers>

</configuration>
```

Spring, JDBC, Mybatis 관련 설정하기 (root-context.xml)

▶ root-context.xml 설정 추가

✓ DBCP(커넥션풀) 설정 추가

```
<!-- DBCP 사용을 위한 DataSource를 Bean으로 등록 -->
<!-- DataSource란? : java에서 Connection Pool을 지원하기 위한 인터페이스 -->
<!-- BasicDataSource : DataSource인터페이스를 구현한 클래스, 아파치 commons.dbcp에서 제공 -->
<!-- destroy-method="close" : 주어진 세션을 자동으로 반환(close)하라는 설정 -->
<bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource" destroy-method="close">

	<property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
	<property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />
	<property name="username" value="kh_final" />
	<property name="password" value="kh_final1234" />

		<!-- defaultAutoCommit: SQL 수행 후 자동 COMMIT 설정. (기본값 : true) -->
	<property name="defaultAutoCommit" value="false" />

	<!-- 커넥션 풀 설정 -->
	<property name="initialSize" value="10" /> <!-- 초기 커넥션 수, 기본 0 -->
	<property name="maxTotal" value="500" /> <!-- 최대 커넥션 수, 기본 8 -->
	<property name="maxIdle" value="100" /> <!-- 유휴 상태로 존재할 수 있는 커넥션 최대 수, 기본 8 -->
	<property name="minIdle" value="10" /> <!-- 유휴 상태로 존재할 수 있는 커넥션 최소 수, 기본 0 -->
	<property name="maxWaitMillis" value="-1" /> <!-- 예외 발생 전 커넥션이 반환 될 때 까지 대기하는 최대 시간(ms), 기본 -1(무기한) -->

</bean>
```

▶ root-context.xml 설정 추가

✓ SqlSession 관련 설정 및 트랜잭션 제어 설정 추가

```
<!-- SqlSession : sql구문을 DB에 전달, 실행하는 객체
SqlSessionFactory : SqlSession을 만드는 객체
sqlSessionFactoryBean : mybatis 설정 파일(mybatis-config.xml)과 Connection Pool 정보를 이용하여 SqlSessionFactory를 만드는 객체
sqlSessionTemplate : SqlSession 객체에 트랜잭션 처리 역할이 가능하도록 하는 객체 -->

<!-- 마이바티스 SqlSession 등록하기 (xml 방식으로 bean 등록) -->
<bean id="sqlSessionFactoryBean" class="org.mybatis.spring.SqlSessionFactoryBean">
	<!-- mybatis-config.xml 설정 불러오기 -->
	<property name="configLocation" value="classpath:mybatis-config.xml" />
	<property name="dataSource" ref="dataSource" />
</bean>

<!-- SqlSessionTemplate : 기본 SQL 실행 + 트랜잭션 관리 역할을 하는 SqlSession을 생성할 수 있게 하는 객체(Spring bean으로 등록해야함.) -->
<bean id="sqlSessionTemplate" class="org.mybatis.spring.SqlSessionTemplate">
		<constructor-arg ref="sqlSessionFactoryBean" />
</bean>

<!-- 스프링에서 사용하는 proxy를 이용한 트랜잭션 제어가 안될 경우 추가적인 트랜잭션 매니저를 추가해서 문제 해결 -->
<bean id="transactionManager" class="org.springframework.jdbc.datasource.DataSourceTransactionManager">
		<property name="dataSource" ref="dataSource" />
</bean>
```

▶ mybatis-config 설정하기 - 참고

✓ Mybatis 내장 별칭

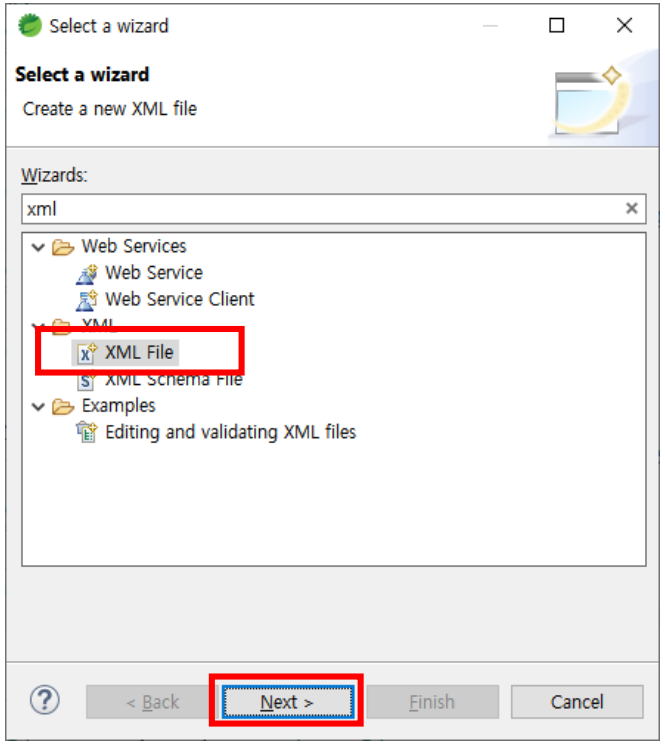
Mybatis 타입	Java 자료형	Mybatis 타입	Java 자료형
_byte	byte	double	Double
_long	long	float	Float
_short	short	boolean	Boolean
_int / _integer	int	date	Date
_double	double	object	Object
_float	float	map	Map
_boolean	boolean	hashmap	HashMap
string	String	list	List
byte	Byte	arraylist	ArrayList
long	Long	collection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

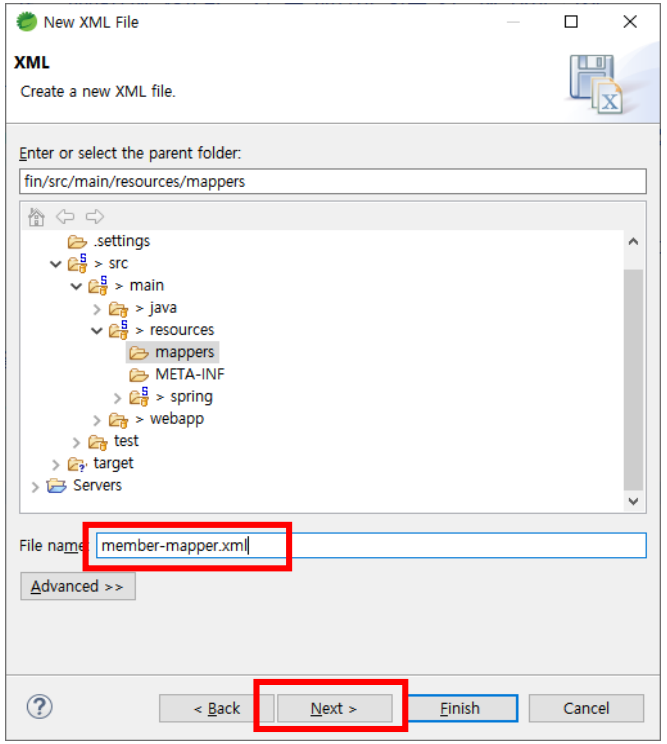
mapper 설정하기

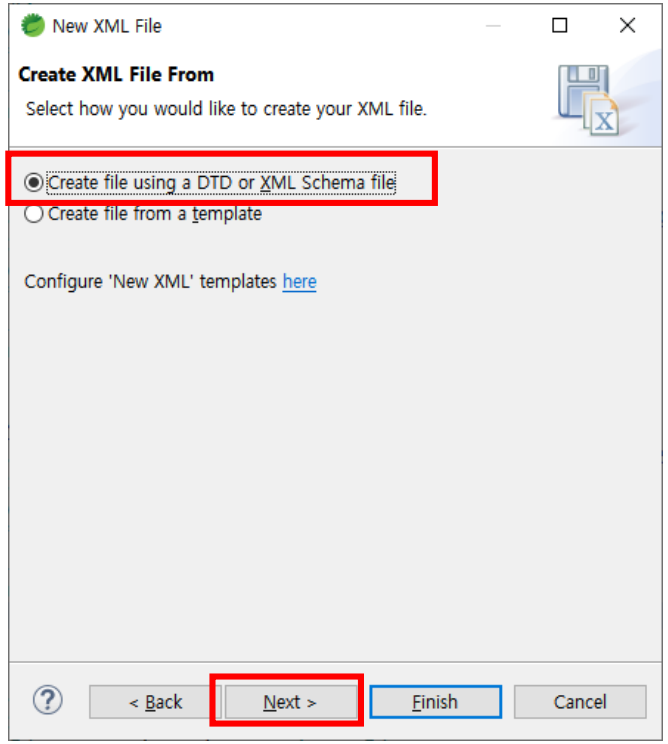
▶ mapper 설정하기

✓ *-mapper.xml 생성 위치

'**src/main/resources**'폴더에 '**mappers**' 폴더 생성 후 mapper.xml 파일 생성



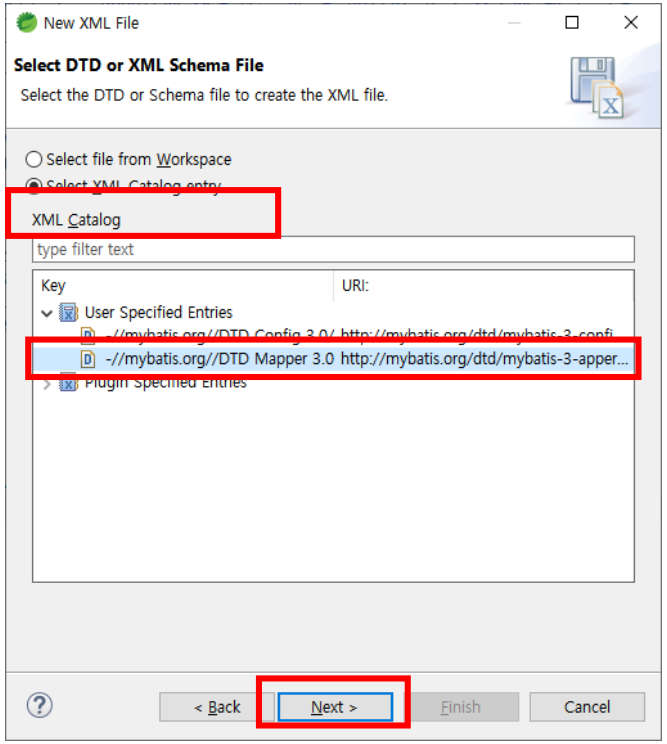


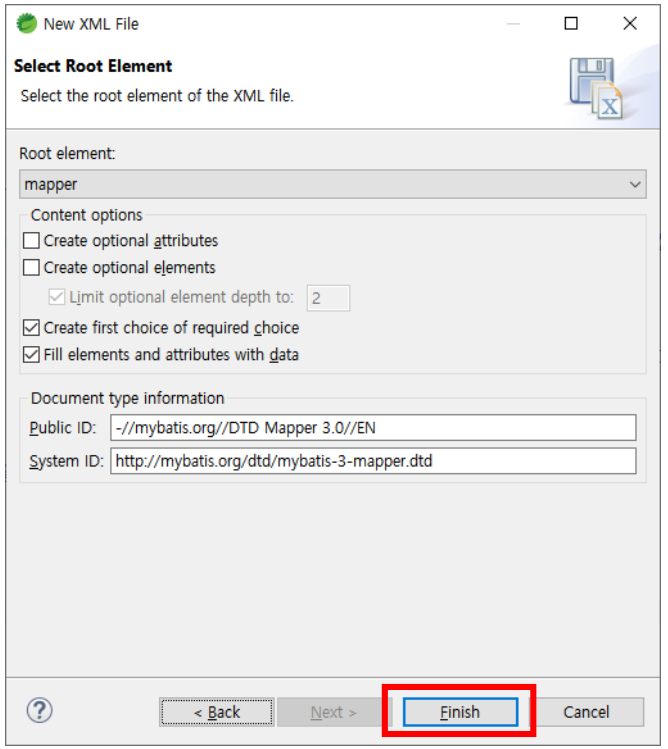


▶ mapper 설정하기

✓ *-mapper.xml 생성 위치

'**src/main/resources**'폴더에 '**mappers**' 폴더 생성 후 mapper.xml 파일 생성





▶ mapper 설정하기

✓ *-mapper.xml 작성

- **<resultMap> 태그 :**
  SELECT 조회 결과(ResultSet)의 컬럼명과 VO객체의 필드명이 일치하지 않을 때
  어떤 컬럼과 필드가 매칭 되어야 하는지 지정하는 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

- 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 에러를 유발한다. -->
    SELECT *
    FROM MEMBER
    WHERE USER_ID = #{userId}
</select>
```

** MyBatis에 작성되는 SQL에서 #{}, ${} 의 의미

- #{ } : PrepareStatement의 위치홀더(?)의 역할로 전달된 값을 의미함.

ex) value = **100**; 인 경우 **#{value} == 100** , value = **"apple"**; 인 경우 **#{value} == 'apple'**

- ${ } : Statement의 역할로 SQL 구문 자체를 의미함.

ex) value = "MEMBER_ID" 인 경우 SELECT **${value}** FROM MEMBER;

== SELECT **MEMBER_ID** FROM MEMBER;

▶ mapper 설정하기 - 참고

✓ <select> 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
resultType	리턴되는 타입의 패키지 경로를 포함한 전체 클래스 명이나 별칭, collection인 경우 list, arraylist로 설정할 수 있다.
resultMap	사용할 resultMap의 id를 기술한다.

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

▶ mapper 설정하기

✓ *-mapper.xml 작성

파라미터로 객체를 받는 경우 해당 객체의 필드명을
작성하는 것을 값을 얻어올 수 있다.

*** <insert>, <update>, <delete> 태그 : 해당 태그들은 설정이 동일**

- <insert> 태그 예시

```
<insert id="insertMember" parameterType="Member" flushCache="true"
        statementType="PREPARED", useGeneratedKeys="true" timeout="20">

    INSERT INTO MEMBER VALUES(
        #{userId}, #{userPwd}, #{userName}
    )

</insert>
```

** DML 구문은 SQL 수행이 성공된 행의 개수를 반환하기 때문에
별도의 resultType, resultMap 속성을 작성하지 않음.

▶ mapper 설정하기

✓ *-mapper.xml 작성

* **<insert>, <update>, <delete> 태그 : 해당 태그들은 설정이 동일**

- <update> 태그 예시

```
<update id="updateMember" parameterType="Member" flushCache="true"
        statementType="PREPARED" timeout="20">
    UPDATE MEMBER
    SET USER_PWD = #{userPwd}, USER_NAME = #{userName}
    WHERE USER_ID = #{userId}
</update>
```

- <delete> 태그 예시

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

▶ 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.