mybatis

1. Persistence Framework?
   1)Persistence :
      - 지속성을 의미, 즉 애플리케이션을 종료하고 다시 실행하더라도 이전에 저장한 데이터를 다시 불러올 수 있는 기술
   2)Persistence Framework
      - 데이터의 저장, 조회, 변경, 삭제를 다루는 클래스 및 설정 파일들의 집합
   3)종류
      -SQL Mapper : SQL 문장으로 직접 DB 데이터를 다루는 기술, mybatis 가 대표
      -Object-Relational Mapper : Java 객체를 통해 간접적으로 DB 데이터를 다루는 기술, Hibernate와 TopLink가 대표

2. 구조
   1)Java Object와 SQL문 사이의 자동 Mapping 기능을 지원하는 ORM Framework이다.
   2)MyBatis는 SQL을 별도의 파일로 분리하여 관리하게 해 주며, 객체-SQL 사이의 파라미터 Mapping 작업을 자동으로 해주기 때문에 많은 인기를 얻고 있는 기술이다.
   3)Hibernate나 JPA처럼 새로운 DB 프로그래밍 패러다임을 익혀야 하는 부담이 없이, 개발자가 익숙한 SQL을 그대로 이용하면서 JDBC 코드 작성의 불편함도 제거해주고, 도메인 객체나 VO 객체를 중심으로 개발이 가능하다는 장점이 있다.
   4)데이터 처리를 위해 DAO는 mybatis에 제공하는 객체의 메소드를 호출
   5)mybatis는 SQL문이 저장된 Mapper 파일에서 데이터 처리에 필요한 SQL 문을 찾는다.
   6)mybatis는 Mapper 파일에서 찾은 SQL을 서버에 보내고자 JDBC 드라이버를 사용한다.
   7)JDBC 드라이버는 SQL문을 데이터베이스 서버로 보낸다.
   8)mybatis는 SELECT 문의 실행 결과를 값 객체에 담아서 반환한다.
   9)INSERT, UPDATE, DELETE 문인 경우 입력, 변경, 삭제된 레코드의 갯수를 반환한다.

3. 특징
   1)쉬운 접근성과 코드의 간결함
      -가장 간단한 Persistence Framework이다.
      -XML 형태로 서술된 JDBC 코드라고 생각해도 될 만큼 JDBC의 모든 기능을 MyBatis가 대부분 제공한다.
      -복잡한 JDBC 코드를 걷어내며 깔끔한 소스코드를 유지할 수 있다.
      -수동적인 파라미터 설정과 쿼리 결과에 대한 맵핑 구문을 제거할 수 있다.
   2)SQL문과 프로그래밍 코드의 분리
      -SQL변경이 있을 때마다 자바 코드를 수정하거나 컴파일하지 않아도 된다.
      -SQL 작성과 관리 또는 검토를 DBA와 같은 개발자가 아닌 다른 사람에게 맡길 수도 있다.

4. Architecture
   1) http://terasolunaorg.github.io/guideline/5.0.0.RELEASE/en/ArchitectureInDetail/DataAccessMyBatis3.html 그림참조

5. MyBatis의 주요 컴포넌트의 역할
   1)MyBatis 설정파일(SqlMapConfig.xml) : Database의 접속 주소 정보나 Mapping 파일의 경로 등의 고정된 환경정보를 설정
   2)SqlSessionFactoryBuilder : MyBatis 설정 파일을 바탕으로 SqlSessionFactory를 생성
   3)SqlSessionFactory : SqlSession을 생성
   4)SqlSession : 핵심적인 역할을 하는 클래스로서 SQL 실행이나 트랜잭션 관리를 실행. SqlSession Object는 Thread-Safe하지 않기 때문에 Thread마다 필요에 따라 생성
   5)mapping File : SQL문과 OR Mapping을 설정.

6. 준비
   1) mariaDB 64bit or Oracle 64bit
   2) https://github.com/mybatis
   3) 프로젝트 목록에서 'mybatis-3' 클릭 (https://github.com/mybatis/mybatis-3)
   4) README.md 에서 Essentials > Download Latest (https://github.com/mybatis/mybatis-3/releases)
   5) Download mybatis-3.4.5.zip
   6) WEB-INF/lib/에 mybatis-3.4.5.jar, mariadb-java-client-2.2.0.jar 를 복사
   6) J2SE이면 Build-path에 추가

7. Test : MySQL의 World Database의 City Table 가져오기
   1)com.javasoft.libs.CityInfo.java
      package com.javasoft.libs;

```
public class CityInfo {
   private int id;
   private String name;
   private String countryCode;
   private String district;
   private int population;
   public int getId() {
      return id;
   }
   public void setId(int id) {
      this.id = id;
   }
   public String getName() {
      return name;
   }
   public void setName(String name) {
      this.name = name;
   }
   public String getCountryCode() {
      return countryCode;
   }
   public void setCountryCode(String countryCode) {
      this.countryCode = countryCode;
   }
   public String getDistrict() {
      return district;
   }
   public void setDistrict(String district) {
      this.district = district;
   }
   public int getPopulation() {
      return population;
   }
   public void setPopulation(int population) {
      this.population = population;
   }
}
```

2)src/mybatis-config.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>
   <properties resource="dbinfo.properties" />
   <typeAliases>
      <typeAlias type="com.javasoft.libs.CityInfo" alias="CityInfo" />
   </typeAliases>

   <environments default="development">
      <environment id="development">
         <transactionManager type="JDBC"/>
         <dataSource type="POOLED">
            <property name="driver" value="org.mariadb.jdbc.Driver"/>
            <property name="url" value="jdbc:mariadb://localhost:3306/world"/>
            <property name="username" value="root"/>
            <property name="password" value="javamariadb"/>
         </dataSource>
      </environment>
   </environments>
   <mappers>
      <mapper resource="com/javasoft/libs/mybatis-mapper.xml"/>
   </mappers>
</configuration>
```

3)com.javasoft.libs.mybatis-mapper.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="com.javasoft.libs.CityInfo">
   <resultMap id="cityInfoResult" type="CityInfo">
      <result property="id" column="ID" />
      <result property="name" column="Name" />
      <result property="district" column="District" />
      <result property="countryCode" column="CountryCode" />
      <result property="population" column="Population" />
   </resultMap>
   <select id="selectInfo" parameterType="int" resultType="cityInfo"
   resultMap="cityInfoResult">
      SELECT * FROM world.city WHERE id = #{id}
   </select>

</mapper>
```

4)com.javasoft.libs.SelectOne.java

```
package com.javasoft.libs;

import java.io.IOException;
import java.io.Reader;

import org.apache.ibatis.io.Resources;
import org.apache.ibatis.session.SqlSession;
import org.apache.ibatis.session.SqlSessionFactoryBuilder;

public class SelectOne {
   public static void main(String[] args) throws IOException {
      String conf = "com/javasoft/libs/mybatis-config.xml";
      Reader reader = Resources.getResourceAsReader(conf);
      SqlSession session = new SqlSessionFactoryBuilder().build(reader).openSession();

      try{
         CityInfo cityInfo = (CityInfo)session.selectOne("selectInfo", 1);

         int id = cityInfo.getId();
         String name = cityInfo.getName();
         String district = cityInfo.getDistrict();
         String code = cityInfo.getCountryCode();
         int population = cityInfo.getPopulation();

         System.out.println("ID : " + id);
         System.out.println("name : " + name);
         System.out.println("District : " + district);
         System.out.println("code : " + code);
         System.out.println("Population : " + population);
      }finally{
         session.close();
      }
   }
}
```

5)com.javasoft.libs.mybatis-mapper.xml

```
<select id="selectListInfo" resultType="CityInfo" resultMap="cityInfoResult">
   SELECT * FROM World.City ORDER BY id DESC
</select>
```

6)com.javasoft.libs.Select.java

```
package com.javasoft.libs;

import java.io.IOException;
import java.io.Reader;
import java.util.List;

```

```
194     import org.apache.ibatis.io.Resources;
195     import org.apache.ibatis.session.SqlSession;
196     import org.apache.ibatis.session.SqlSessionFactoryBuilder;
197
198     public class Select {
199        public static void main(String[] args) throws IOException {
200           String conf = "com/javasoft/libs/mybatis-config.xml";
201           Reader reader = Resources.getResourceAsReader(conf);
202           SqlSession session = new SqlSessionFactoryBuilder().build(reader).openSession();
203           try{
204              List<CityInfo> list = session.selectList("selectListInfo");
205
206              for(CityInfo vo : list){
207                 System.out.println(vo.getId());
208                 System.out.println(vo.getName());
209                 System.out.println(vo.getDistrict());
210                 System.out.println(vo.getCountryCode());
211                 System.out.println(vo.getPopulation());
212                 System.out.println("-----------------------");
213              }
214           }finally{
215              session.close();
216           }
217        }
218     }
219
220  7)com.javasoft.libs.mybatis-mapper.xml
221     <update id="updateCityInfo" parameterType="CityInfo">
222        UPDATE world.city SET name = #{name}, countrycode = #{countryCode}, district =
        #{district}
223                              , population = #{population}
224        WHERE id = #{id}
225  </update>
226
227  8)com.javasoft.libs.Update.java
228     package com.javasoft.libs;
229
230     import java.io.IOException;
231     import java.io.Reader;
232
233     import org.apache.ibatis.io.Resources;
234     import org.apache.ibatis.session.SqlSession;
235     import org.apache.ibatis.session.SqlSessionFactoryBuilder;
236
237     public class Update {
238        public static void main(String[] args) throws IOException {
239           String conf = "com/javasoft/libs/mybatis-config.xml";
240           Reader reader = Resources.getResourceAsReader(conf);
241           SqlSession session = new SqlSessionFactoryBuilder().build(reader).openSession();
242
243           try{
244              CityInfo city = new CityInfo();
245              city.setId(2362);  //Yong-in
246              city.setName("Kiheung");
247              city.setCountryCode("KOR");
248              city.setDistrict("Kyonggi");
249              city.setPopulation(500000);
250
251              session.update("updateCityInfo", city);
252              session.commit();
253              System.out.println("Update Success");
254           }finally{
255              session.close();
256           }
257        }
258     }
259
```

8. 시나리오
  1) MySqlProjectDao는 SqlSessionFactory 에게 SQL을 실행할 객체를 요구
  2) SqlSessionFactory는 SqlSession 객체를 생성하여 반환
  3) MySqlProjectDao는 SqlSession 객체에게 SQL 실행을 요청
  4) SqlSession 객체는 SQL 이 저장된 맵퍼 파일에서 SQL을 찾는다.
  5) SqlSession 은 JDBC 드라이버를 통해 데이터베이스에 질의를 실행한다.
  6) SqlSession 은 데이터베이스로부터 가져온 데이터로 Project 목록을 생성하여 반환
  7) MySqlProjectDao는 사용이 끝난 SqlSession 을 닫는다.