1. MyBatis 개요
   1)MyBatis(http://www.mybatis.org/mybatis-3)는 Java Object와 SQL 문 사이의 자동 Mapping 기능을 지원하는 ORM 프레임워크이다.
   2)MyBatis는 SQL을 별도의 파일로 분리해서 관리하게 해주며, 객체-SQL 사이의 파라미터 Mapping 작업을 자동으로 해주기 때문에 많은 인기를 얻고 있는 기술
   3)MyBatis는 Hibernate나 JPA(Java Persistence API)처럼 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.

2. 특징
   1)쉬운 접근성과 코드의 간결함
      -가장 간단한 persistence 프레임워크이다.
      -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)SQL 문과 프로그래밍 코드의 분리
      -SQL에 변경이 있을 때마다 자바 코드를 수정하거나 컴파일 하지 않아도 된다.
      -SQL 작성과 관리 또는 검토를 DBA와 같은 개발자가 아닌 다른 사람에게 맡길 수도 있다.

   3)다양한 프로그래밍 언어로 구현가능
      -Java, C#, .NET, Ruby

3. MyBatis와 MyBatis-Spring을 사용한 DB Access Architecture
   -Application Modules
      --Service --> Repository(Mapper)
   -O/R Mapper
      --MyBatis 3
      --MyBatis-Spring
   -JDBC Intefaces
      --JDBC Basic APIs
      --DataSource(Configuration for Connect)
   -JDBC Implementations
      --JDBC Driver
   -Persistence Layer
      --Database

4. MyBatis를 사용하는 Data Access Layer
   1)Presentation Layer
      -Controller
      -MultiActionController

   2)Service Layer
      -Service
      -ServiceImpl

   3)Data Access Layer
      -Dao
      -DaoImpl

   4)MyBatis Framework

-mapper.xml
-jdbc.properties
-sqlMapConfig.xml
-SqlSessionFactory
-SqlSession

5. MyBatis 3의 주요 컴포넌트
-그림참조
-http://terasolunaorg.github.io/guideline/5.0.0.RELEASE/en/ArchitectureInDetail/DataAccessMyBatis3.html

6. MyBatis 3의 주요 컴포넌트의 역할
1)MyBatis 설정파일(SqlMapconfig.xml)
-데이터베이스의 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 오브젝트는 Thread-Safe하지 않으므로 thread마다 필요에 따라 생성
5)Mapping File(user.xml)
-SQL문과 ORMapping을 설정

7. MyBatis-Spring의 주요 컴포넌트의 역할
1)MyBatis 설정파일(sqlMapConfig.xml)
-VO 객체의 정보를 설정
2)SqlSessionFactoryBean
-MyBatis 설정파일을 바탕으로 SqlSessionFactory를 생성
-Spring Bean으로 등록해야 함.
3)SqlSessionTemplate
-핵심적인 역할을 하는 클래스로서 SQL 실행이나 트랜잭션 관리를 실행한다.
-SqlSession 인터페이스를 구현하며, Thread-Safe하다.
-Spring Bean으로 등록해야 함.
4)Mapping File(mybatis-mapper.xml)
-SQL문과 OR Mapping을 설정
5)Spring Bean 설정파일(beans.xml)
-SqlSessionFactoryBean을 Bean 등록할 때 DataSource 정보와 MyBatis Config 파일정보, Mapping 파일의 정보를 함께 설정한다.
-SqlSessionTemplate을 Bean으로 등록한다.

8. Lab : MySQL의 World Database의 City Table 가져오기
1)MariaDB 설치하기
-https://mariadb.org
-https://downloads.mariadb.org에서 MariaDB 10.3.15 Stable
-mariadb-10.3.15-winx64.msi

2)MyBatisDemo project 생성
-In Package Explorer > right-click > New > Java Project
-Project name : MybatisDemo

```
103
104   3)src > right-click > New > Package
105     -Package name : com.example
106
107   4)Java Project를 Spring Project로 변환
108     -MybatisDemo Project > right-click > Configuration > Convert to Maven Project
109     -Project : /MybatisDemo
110     -Group Id : MybatisDemo
111     -Artifact Id : MybatisDemo
112     -version : 0.0.1-SNAPSHOT
113     -Packaging : jar
114     -Finish
115     -Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.
116
117     -MybatisDemo Project > right-click > Spring > Add Spring Project Nature
118     -Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.
119
120     -pom.xml 파일에 Spring Context Dependency 추가하기
121       <version>0.0.1-SNAPSHOT</version>
122       <dependencies> <--- dependencies element 추가
123         <dependency> <---여기에 paste
124           <groupId>org.springframework</groupId>
125           <artifactId>spring-context</artifactId>
126           <version>4.3.24.RELEASE</version>
127         </dependency>
128       </dependencies>
129
130     -pom.xml > right-click > Run As > Maven install
131       [INFO] BUILD SUCCESS 확인
132
133   5)MyBatis library 검색 및 설치
134     -Maven Repositoy에서 'mybatis'로 검색
135
136       <dependency>
137         <groupId>org.mybatis</groupId>
138         <artifactId>mybatis</artifactId>
139         <version>3.5.1</version>
140       </dependency>
141
142     -pom.xml에 등록 및 설치
143
144   6)MyBatis-Spring library 검색 및 설치
145     -Maven Repository에서 'mybatis spring'으로 검색
146
147       <dependency>
148         <groupId>org.mybatis</groupId>
149         <artifactId>mybatis-spring</artifactId>
150         <version>2.0.1</version>
151       </dependency>
152
153     -pom.xml에 등록 및 설치
154
155   7)MariaDB Jdbc Driver 라이브러리 검색 및 설치
156     -Maven Repository 에서 'mariadb'로 검색하여 MariaDB Java Client를 설치한다.
```

```
157
158         <dependency>
159             <groupId>org.mariadb.jdbc</groupId>
160             <artifactId>mariadb-java-client</artifactId>
161             <version>2.4.1</version>
162         </dependency>
163
164     -pom.xml에 붙여 넣고 Maven Install 하기
165
166   8)Spring JDBC 설치
167     -JdbcTemplate를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.
168
169         <dependency>
170           <groupId>org.springframework</groupId>
171           <artifactId>spring-jdbc</artifactId>
172           <version>4.3.24.RELEASE</version>
173         </dependency>
174
175     -pom.xml에 붙여 넣고 Maven Install 하기
176
177   9)jUnit Library 설치
178     -http://mvnrepository.com에 접근
179     -jUnit으로 검색
180     -jUnit 4.12 버전을 pom.xml에 추가
181
182       <dependency>
183         <groupId>junit</groupId>
184         <artifactId>junit</artifactId>
185         <version>4.12</version>
186         <scope>test</scope>
187       </dependency>
188
189     -pom.xml > right-click > Run As > Maven Install
190
191   10)MybatisDemo/resources folder 생성
192     -MybatisDemo project > right-click > Build Path > Configure Build Path
193     -Source Tab > Add Folder
194     -MybatisDemo click
195     -Create New Folder > Folder name : resources > Finish > OK
196     -MybatisDemo/resources(new) 확인
197     -Apply and Close
198
199   11)resources/dbinfo.properties 파일 생성
200     -/resoures/dbinfo.properties 파일 생성
201     -/resources > right-click > New > File
202     -File name : dbinfo.properties > Finish
203
204       db.driverClass=org.mariadb.jdbc.Driver
205       db.url=jdbc:mariadb://localhost:3306/world
206       db.username=root
207       db.password=javamariadb
208
209   12)world database downloads
210     -https://dev.mysql.com/doc/index-other.html
```

-Example Databases > world database > Zip
-Unzip
-MariaDB login 후 world database 실행

13)여러 Package 생성
-/src/com.example.vo
-/src/com.example.service
-/src/com.example.dao

14)VO 클래스 작성
-/src/com.example.vo.CityVO.java 생성

```
package com.example.vo;

public class CityVO {
  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;
  }
  @Override
  public String toString() {
    return String.format("CityInfoVO [id=%s, name=%s, countryCode=%s, district=%s,
    population=%s]", id, name,
```

```
                countryCode, district, population);
        }
      }

  15)Mapping 파일 작성 및 MyBatis 설정
    -/resources/SqlMapConfig.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>
          <typeAliases>
            <typeAlias type="com.example.vo.CityVO" alias="cityVO" />
          </typeAliases>

        </configuration>

    -/resources/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="City">
        <resultMap id="cityResult" type="cityVO">
          <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="selectCityByName" parameterType="String" resultType="cityVO"
        resultMap="cityResult">
          SELECT * FROM world.city WHERE  name = #{name}
        </select>
      </mapper>

  16)Dao 객체 생성
    -/src/com.example.dao.CityDao.java

      package com.example.dao;

      import java.util.List;
      import com.example.vo.CityVO;

      public interface CityDao {
        List<CityVO> readAll();
        CityVO read(String name);
      }

    -/src/com.example.dao.CityDaoImpl.java
```

```
317         package com.example.dao;
318
319         import java.util.List;
320
321         import org.apache.ibatis.session.SqlSession;
322         import org.springframework.beans.factory.annotation.Autowired;
323         import org.springframework.stereotype.Repository;
324
325         import com.example.vo.CityVO;
326         @Repository("cityDao")
327         public class CityDaoImpl implements CityDao {
328
329           @Autowired
330             private SqlSession session;
331
332           @Override
333           public List<CityVO> readAll() {
334             return null;
335           }
336
337           @Override
338           public CityVO read(String name) {
339             CityVO city  = session.selectOne("City.selectCityByName", name);
340             return city;
341           }
342         }
343
344     17)Service 객체 생성
345       -/src/com.example.service.CityService.java
346
347           package com.example.service;
348
349           import java.util.List;
350           import com.example.vo.CityVO;
351
352           public interface CityService {
353             List<CityVO> getCityList();
354             CityVO getCity(String name);
355           }
356
357       -/src/com.example.service.CityServiceImpl.java
358           package com.example.service;
359
360           import java.util.List;
361
362           import org.springframework.beans.factory.annotation.Autowired;
363           import org.springframework.stereotype.Service;
364
365           import com.example.dao.CityDao;
366           import com.example.vo.CityVO;
367
368           @Service("cityService")
369           public class CityServiceImpl implements CityService {
370
```

```
371             @Autowired
372             CityDao cityDao;
373
374             @Override
375             public List<CityVO> getCityList() {
376               return null;
377             }
378
379             @Override
380             public CityVO getCity(String name) {
381               return this.cityDao.read(name);
382             }
383           }
384
385     18)Bean Configuration XML 작성
386       -/resources > right-click > New > Spring Bean Configuration File
387       -File name : beans.xml > Finish
388       -Namespace Tab
389       -Check context - http://www.springframework.org/schema/context
390
391         <?xml version="1.0" encoding="UTF-8"?>
392         <beans xmlns="http://www.springframework.org/schema/beans"
393           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
394           xmlns:context="http://www.springframework.org/schema/context"
395           xsi:schemaLocation="http://www.springframework.org/schema/beans
              http://www.springframework.org/schema/beans/spring-beans.xsd
396             http://www.springframework.org/schema/context
                http://www.springframework.org/schema/context/spring-context-3.2.xsd">
397
398         <!-- mybatis-spring 설정 -->
399         <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
400           <property name="dataSource"  ref="dataSource" />
401           <property name="configLocation"   value="classpath:SqlMapConfig.xml" />
402           <property name="mapperLocations">
403             <list>
404               <value>classpath:mybatis-mapper.xml</value>
405             </list>
406           </property>
407         </bean>
408
409         <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
410           <constructor-arg ref="sqlSessionFactory" />
411         </bean>
412
413         <context:property-placeholder location="classpath:dbinfo.properties" />
414         <bean id="dataSource"
          class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
415           <property name="driverClass" value="${db.driverClass}" />
416           <property name="url" value="${db.url}" />
417           <property name="username" value="${db.username}" />
418           <property name="password" value="${db.password}" />
419         </bean>
420       </beans>
421
```

19)사용자 관리 프로젝트의 Bean 등록 및 의존 관계 설정
-<context:component-scan> 태그 사용
-@Service, @Repository 어노테이션을 선언한 클래스들과 @Autowired 어노테이션을 선언하여 의존관계를 설정한 클래스들이 위치한 패키지를 Scan하기 위한 설정을 XML에 해주어야 한다.
-beans.xml에 다음 코드 추가한다.

```
<context:component-scan base-package="com.example" />
```

20)Spring TestContext Framework 사용하기
-/src/com.example.test package 생성
-/src/com.example.test > right-click > New > JUnit Test Case
-Name : MybatisDemoTest > Finish
-New JUnit Test Case차에서 [Not now] 선택 > OK

```
import org.junit.Test;
import org.junit.runner.RunWith;
import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.test.context.ContextConfiguration;
import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

import com.example.service.CityService;
import com.example.vo.CityVO;

@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations="classpath:beans.xml")
public class MybatisDemoTest {
  @Autowired
  CityService cityService;

  @Test
  public void test() {
    CityVO city = cityService.getCity("Seoul");
    System.out.println(city);
  }
}
```

21)Test
-right-click > Run As > Junit Test
-결과 -> Junit View에 초록색 bar
CityInfoVO [id=2331, name=Seoul, countryCode=KOR, district=Seoul, Poplulation=9981619]

22)All City 읽어오기
-com.example.service.CityServiceImpl.java

```
@Override
public List<CityVO> getCityList() {
  return citydao.readAll();
}
```

-com.example.dao.CityDaoImpl.java

```
@Override
public List<CityVO> readAll() {
```

```
474         List<CityVO> cityList = session.selectList("City.selectList");
475         return cityList;
476       }
477
478     -mybatis-mapper.xml
479
480       <select id="selectList" resultType="cityVO" resultMap="cityResult">
481         SELECT * FROM world.city ORDER BY id DESC
482       </select>
483
484     -MybatisDemoTest.java
485
486       @Autowired
487       CityService cityService;
488
489       @Ignore @Test
490       public void test() {
491         CityVO city = this.cityService.getCity("Seoul");
492         System.out.println(city);
493       }
494
495       @Test
496       public void test1() {
497         List<CityVO> list = this.cityService.getCityList();
498
499         for(CityVO vo : list){
500           System.out.println(vo.getId());
501           System.out.println(vo.getName());
502           System.out.println(vo.getDistrict());
503           System.out.println(vo.getCountryCode());
504           System.out.println(vo.getPopulation());
505           System.out.println("-----------------------");
506         }
507       }
508
509   23)Test
510     -right-click > Run As > Junit Test
511     -결과 -> Junit View에 초록색 bar
```