1 1. MyBatis 개요

- 2 **1)MyBatis(http://www.mybatis.org/mybatis-3)**는 Java Object와 **SQL** 문 사이의 자동 **Mapping** 기능을 지원하는 **ORM** 프레임워크이다.
- 3 2)MyBatis는 SQL을 별도의 파일로 분리해서 관리하게 해주며, 객체-SQL 사이의 파라미터 Mapping 작업을 자동으로 해주기 때문에 많은 인기를 얻고 있는 기술
- 4 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.

5 6 7

8

9

10

- 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.
- 11 -복잡한 JDBC 코드를 걷어내며 깔끔한 소스코드를 유지할 수 있다.
 - -수동적인 파라미터 설정과 쿼리 결과에 대한 맵핑 구문을 제거할 수 있다.

12 13 14

15

16

17 18

- 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

19 20 21

22

24

26

27

28

29

30

31

32

33

34

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

35 36 37

38

39

40

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

41 42 43

- 2)Service Layer
 - -Service
 - -ServiceImpl

44 45 46

47

- 3)Data Access Layer
- -Dao
- -DaoImpl

48 49 50

51

52 53

54

- 4)MyBatis Framework
 - -mapper.xml
 - -jdbc.properties
 - -sqlMapConfig.xml
 - -SqlSessionFactory
 - -SqlSession

55 56 57

58

60

- 5. MyBatis 3의 주요 컴포넌트
- 59 그림참조

-http://terasolunaorg.github.io/guideline/5.0.0.RELEASE/en/ArchitectureInDetail/DataAccessMyBatis 3.html

```
62
 63
    6. MyBatis 3의 주요 컴포넌트의 역할
 64
       1)MyBatis 설정파일(SqlMapconfig.xml)
 65
         -데이터베이스의 접속 주소 정보나 Mapping 파일의 경로 등의 고정된 환경정보를 설정
 66
       2)SqlSessionFactoryBuilder
         -MyBatis 설정 파일을 바탕으로 SqlSessionFactory를 생성
 67
 68
       3)SqlSessionFactory
 69
         -SqlSession을 생성
 70
       4)SqlSession
 71
         -핵심적인 역할을 하는 클래스로서 Sql 실행이나 트랜잭션 관리를 실행
 72
         -SqlSession 오브젝트는 Thread-Safe하지 않으므로 thread마다 필요에 따라 생성
 73
       5) Mapping File(user.xml)
 74
         -SQL문과 ORMapping을 설정
 75
 76
 77
     7. MyBatis-Spring의 주요 컴포넌트의 역할
 78
       1)MyBatis 설정파일(sqlMapConfig.xml)
 79
         -VO 객체의 정보를 설정
 80
       2)SqlSessionFactoryBean
         -MyBatis 설정파일을 바탕으로 SqlSessionFactory를 생성
 81
 82
         -Spring Bean으로 등록해야 함.
 83
       3)SqlSessionTemplate
 84
         -핵심적인 역할을 하는 클래스로서 SQL 실행이나 트랜잭션 관리를 실행한다.
 85
         -SqlSession 인터페이스를 구현하며, Thread-Safe하다.
 86
         -Spring Bean으로 등록해야 함.
 87
       4) Mapping File (mybatis-mapper.xml)
 88
         -SQL문과 OR Mapping을 설정
 89
       5)Spring Bean 설정파일(beans.xml)
 90
         -SqlSessionFactoryBean을 Bean 등록할 때 DataSource 정보와 MyBatis Config 파일정보, Mapping
         파일의 정보를 함께 설정한다.
 91
         -SqlSessionTemplate을 Bean으로 등록한다.
 92
 93
 94
    8. Lab: MySQL의 World Database의 City Table 가져오기
 95
       1)MyBatisDemo project 생성
 96
         -Spring Legacy Project
 97
         -Simple Maven Project
 98
 99
       2)MyBatis library 검색 및 설치
100
         -Maven Repositoy에서 'mybatis'로 검색
101
102
            <dependency>
103
                <groupId>org.mybatis</groupId>
104
                <artifactId>mybatis</artifactId>
105
                <version>3.4.5</version>
106
            </dependency>
107
108
         -pom.xml에 등록 및 설치
109
110
       3)MyBatis-Spring library 검색 및 설치
111
         -Maven Repository에서 'mybatis spring'으로 검색
112
113
            <dependency>
114
                <groupId>org.mybatis</groupId>
115
                <artifactId>mybatis-spring</artifactId>
116
                <version>1.3.1</version>
117
            </dependency>
118
119
         -pom.xml에 등록 및 설치
120
121
       4)MySQL Jdbc Driver 라이브러리 검색 및 설치
122
         -Maven Repository 에서 'spring mysql'로 검색하여 MySQL Connector/J를 설치한다.
123
124
              <dependency>
125
                   <groupId>mysql</groupId>
126
                   <artifactId>mysql-connector-java</artifactId>
127
                   <version>5.1.42</version>
```

```
128
                </dependency>
129
130
          -pom.xml에 붙여 넣고 Maven Install 하기
131
132
        5)Spring JDBC 설치
          -JdbcTemplate를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.
133
134
135
                <dependency>
136
                  <groupId>org.springframework</groupId>
137
                  <artifactId>spring-jdbc</artifactId>
138
                  <version>4.3.9.RELEASE
139
                </dependency>
140
141
          -pom.xml에 붙여 넣고 Maven Install 하기
142
143
        6)dbinfo.properties 파일 생성
          -/src/main/resoures/dbinfo.properties 파일 생성
144
          -/src/main/resources > right-click > New > File
145
146
          -File name : dbinfo.properties > Finish
147
148
             db.driverClass=com.mysql.jdbc.Driver
             db.url=jdbc:mysql://192.168.56.15:3306/world
149
150
             db.username=root
151
             db.password=javamysql
152
        7)여러 Package 생성
153
154
          -/src/main/java/com.javasoft.vo
155
          -/src/main/java/com.javasoft.service
156
          -/src/main/java/com.javasoft.dao
157
158
        8)여러 클래스 작성
159
          -/src/main/java/com.javasoft.CityVO.java 생성
160
161
               package com.javasoft;
162
163
               public class CityVO {
164
                  private int id;
165
                  private String name;
166
                  private String countryCode;
167
                  private String district;
168
                  private int population;
169
                  public int getId() {
170
                     return id;
171
172
                  public void setId(int id) {
173
                     this.id = id;
174
175
                  public String getName() {
176
                     return name;
177
178
                  public void setName(String name) {
179
                     this.name = name;
180
181
                  public String getCountryCode() {
182
                     return countryCode;
183
184
                  public void setCountryCode(String countryCode) {
185
                     this.countryCode = countryCode;
186
187
                  public String getDistrict() {
188
                     return district;
189
190
                  public void setDistrict(String district) {
191
                     this.district = district;
192
193
                  public int getPopulation() {
194
                     return population;
```

```
195
                   }
196
                   public void setPopulation(int population) {
197
                      this.population = population;
198
199
                   @Override
                   public String toString() {
200
                      return String.format("CityInfoVO [id=%s, name=%s, countryCode=%s, district=%s,
201
                      population=%s]", id, name,
202
                           countryCode, district, population);
203
                   }
                }
204
205
206
           -/com.javasoft.dao.CityDao.java
207
208
                package com.javasoft.dao;
209
210
                import java.util.List;
211
                import com.javasoft.vo.CityVO;
212
213
                public interface CityDao {
214
                   List<CityVO> readAll();
215
                   CityVO read(String name);
216
                }
217
218
           -/com.javasoft.dao.CityDaoImpl.java
219
220
             package com.javasoft.dao;
221
222
             import java.util.List;
223
224
             import org.apache.ibatis.session.SqlSession;
225
             import org.springframework.beans.factory.annotation.Autowired;
226
             import org.springframework.stereotype.Repository;
227
228
             import com.javasoft.vo.CityVO;
229
             @Repository("cityDao")
230
             public class CityDaoImpl implements CityDao {
231
232
                @Autowired
233
                   private SqlSession session;
234
235
                @Override
236
                public List<CityVO> readAll() {
237
                   return null;
238
                }
239
240
                @Override
241
                public CityVO read(String name) {
242
                   CityVO city = session.selectOne("City.selectCityByName", name);
243
                   return city;
244
             }
245
246
247
           -/com.javasoft.service.CityService.java
248
249
                package com.javasoft.service;
250
251
                import java.util.List;
252
                import com.javasoft.vo.CityVO;
253
254
                public interface CityService {
                   public List<CityVO> getCityList();
255
256
                   public CityVO getCity(String name);
257
                }
258
259
           -/com.javasoft.service.CityServiceImpl.java
260
                package com.javasoft.service;
```

```
262
               import java.util.List;
263
264
               import org.springframework.beans.factory.annotation.Autowired;
265
               import org.springframework.stereotype.Service;
266
267
               import com.javasoft.dao.CityDao;
               import com.javasoft.vo.CityVO;
268
269
270
                @Service("cityService")
271
               public class CityServiceImpl implements CityService {
272
273
                  @Autowired
274
                  CityDao citydao;
275
276
                  @Override
277
                  public List<CityVO> getCityList() {
278
                     return citydao.readAll();
279
280
                  @Override
281
282
                  public CityVO getCity(String name) {
283
                     return citydao.read(name);
284
285
               }
286
287
        9)Mapping 파일 작성 및 MyBatis 설정
288
          -/src/main/resources/SqlMapConfig.xml
289
                <?xml version="1.0" encoding="UTF-8" ?>
290
291
                <!DOCTYPE configuration
292
                  PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
293
                     "http://mybatis.org/dtd/mybatis-3-config.dtd">
294
                <configuration>
295
                  <typeAliases>
296
                     <typeAlias type="com.javasoft.vo.CityVO" alias="cityVO" />
                  </typeAliases>
297
298
299
                </configuration>
300
301
          -/src/main/resources/mybatis-mapper.xml
302
303
                <?xml version="1.0" encoding="UTF-8"?>
304
                <!DOCTYPE mapper
305
                  PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
306
                     "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
307
                <mapper namespace="City">
                <resultMap id="cityResult" type="cityVO">
308
309
                  <result property="id" column="ID" />
310
                  <result property="name" column="Name" />
311
                  <result property="district" column="District" />
                  <result property="countryCode" column="CountryCode" />
312
313
                  <result property="population" column="Population" />
314
                </resultMap>
315
316
                <select id="selectCityByName" parameterType="String" resultType="cityVO"</pre>
                resultMap="cityResult">
317
                  SELECT * FROM world.city WHERE name = #{name}
318
                </select>
319
             </mapper>
320
321
        10)Bean Configuration XML 작성
322
          -/src/main/resources > right-click > New > Spring Bean Configuration File
323
          -File name : beans.xml > Finish
324
          -Namespace Tab
325
          -Check context - http://www.springframework.org/schema/context
326
```

261

```
<?xml version="1.0" encoding="UTF-8"?>
327
328
             <beans xmlns="http://www.springframework.org/schema/beans"</p>
329
               xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
330
               xmlns:context="http://www.springframework.org/schema/context"
               xsi:schemaLocation="http://www.springframework.org/schema/beans
331
               http://www.springframework.org/schema/beans/spring-beans.xsd
                  http://www.springframework.org/schema/context
332
                  http://www.springframework.org/schema/context/spring-context-3.2.xsd">
333
334
             <!-- mybatis-spring 설정 -->
             <bean id="sqlSessionFactory" class="org.mybatis.spring.SqlSessionFactoryBean">
335
               cproperty name="dataSource" ref="dataSource" />
336
               configLocation" value="classpath:SqlMapConfig.xml" />
337
               property name="mapperLocations">
338
339
                  <list>
340
                    <value>classpath:mybatis-mapper.xml</value>
341
                  </list>
342
               </property>
343
             </bean>
344
             <bean id="sqlSession" class="org.mybatis.spring.SqlSessionTemplate">
345
346
               <constructor-arg ref="sqlSessionFactory" />
347
             </bean>
348
349
             <context:property-placeholder location="classpath:dbinfo.properties" />
350
             <bean id="dataSource"</pre>
             class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
351
               cproperty name="driverClass" value="${db.driverClass}" />
352
               cproperty name="url" value="${db.url}" />
353
               cproperty name="username" value="${db.username}" />
               property name="password" value="${db.password}" />
354
355
             </bean>
356
          </beans>
357
358
        11)사용자 관리 프로젝트의 Bean 등록 및 의존 관계 설정
359
          -<context:component-scan> 태그 사용
360
          -@Service, @Repository 어노테이션을 선언한 클래스들과 @Autowired 어노테이션을 선언하여
          의존관계를 설정한
361
          클래스들이 위치한 패키지를 Scan하기 위한 설정을 XML에 해주어야 한다.
362
          -beans.xml에 다음 코드 추가한다.
363
364
             <context:component-scan base-package="com.javasoft" />
365
366
        12)Spring TestContext Framework 사용하기
367
          -/src/test/java > right-click > New > JUnit Test Case
368
          -Name : MyBatisDemoTest > Finish
369
370
               import org.junit.Test;
371
               import org.junit.runner.RunWith;
372
               import org.springframework.beans.factory.annotation.Autowired;
373
               import org.springframework.test.context.ContextConfiguration;
374
               import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
375
376
               import com.javasoft.service.CityService;
377
               import com.javasoft.vo.CityVO;
378
379
               @RunWith(SpringJUnit4ClassRunner.class)
380
               @ContextConfiguration(locations="classpath:beans.xml")
381
               public class MyBatisDemoTest {
382
                  @Autowired
383
                  CityService service;
384
385
                  @Test
386
                  public void test() {
387
                    CityVO city = service.getCity("Seoul");
388
                    System.out.println(city);
389
                  }
```

```
390
                }
391
392
        13)All City 읽어오기
393
           -com.javasoft.service.CityServiceImpl.java
394
395
                @Override
396
                public List<CityVO> getCityList() {
397
                   return citydao.readAll();
                }
398
399
400
           -com.javasoft.dao.CityDaoImpl.java
401
402
                @Override
403
                public List<CityVO> readAll() {
404
                   List<CityVO> cityList = session.selectList("City.selectList");
405
                   return cityList;
406
                }
407
408
           -mybatis-mapper.xml
409
             <select id="selectList" resultType="cityVO" resultMap="cityResult">
410
411
                SELECT * FROM world.city ORDER BY id DESC
412
             </select>
413
414
           -MyBatisDemoTest.java
415
416
             @Autowired
417
             CityService service;
418
419
             @Ignore @Test
420
             public void test() {
421
                CityVO city = service.getCity("Seoul");
422
                System.out.println(city);
423
             }
424
             @Test
425
426
             public void test1() {
427
                List < CityVO > list = service.getCityList();
428
429
                for(CityVO vo : list){
430
                   System.out.println(vo.getId());
431
                   System.out.println(vo.getName());
432
                   System.out.println(vo.getDistrict());
433
                   System.out.println(vo.getCountryCode());
434
                   System.out.println(vo.getPopulation());
                   System.out.println("-----");
435
436
                }
437
             }
```