

```
1 HOL : Spring JDBC
2 -----
3 Task1. Spring Jdbc Demo
4 1. SpringJdbcDemo project 생성
5     1)New > Java Project >
6     2)Project name : SpringJdbcDemo > Finish
7     3)JRE
8         -Select [Use default JRE 'jdk-13.0.2' and workspace compiler preferences]
9     4)Next
10    5)Uncheck [Create module-info.java file]
11    6)Finish
12
13
14 2. com.example Package 생성
15     1)src > right-click > New > Package
16     2)Name : com.example
17     3)Finish
18
19
20 3. config folder 생성
21     1)SpringJdbcDemo project > right-click > New > Source Folder
22     2)Folder name : config
23     3)Finish
24
25
26 4. config/dbinfo.properties file 생성
27     1)config > right-click > New > File
28     2)File name : dbinfo.properties
29     3)Finish
30
31     db.driverClass=oracle.jdbc.driver.OracleDriver
32     db.url=jdbc:oracle:thin:@localhost:1521:XE
33     db.username=hr
34     db.password=hr
35
36
37 5. UserClient.java 생성
38     1)src > com.example > right-click > New > Class
39     2)Name : UserClient
40     3)Finish
41
42     public class UserClient{
43         public static void main(String [] args){
44
45         }
46     }
47
48
49 6. Maven Project로 전환
50     1)SpringJdbcDemo Project > right-click > Configure > Convert to Maven Project
51     2)Finish
52
53
54 7. Spring Project로 전환
55     1)SpringJdbcDemo Project > right-click > Spring Tools > Add Spring Project Nature
56
57
58 8. Spring Context 설치
```

1)Maven Repository 에서 'Spring Context'로 검색하여 dependency 추가하고 설치

```
<version>0.0.1-SNAPSHOT</version>
<dependencies>
  <dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.2.5.RELEASE</version>
  </dependency>
</dependencies>
```

9. pom.xml에 Oracle Jdbc Driver 설정하기

1)Oracle 12C 이후 version일 경우

```
<dependency>
  <groupId>com.oracle</groupId>
  <artifactId>ojdbc8</artifactId>
  <version>12.2</version>
</dependency>
```

2)Oracle 11g version일 경우

```
-pom.xml에 붙여 넣고 Maven Install 하기
<dependency>
  <groupId>com.oracle</groupId>
  <artifactId>ojdbc6</artifactId>
  <version>11.2</version>
</dependency>
```

10. Apache DBCP2 pom.xml에 추가하기

1)<https://mvnrepository.com/>에서 'dbcp'으로 검색

2)'Apache Commons DBCP' click

3)2.7.0 click

4)dependency copy해서 pom.xml에 붙여넣기

```
<!-- https://mvnrepository.com/artifact/org.apache.commons/commons-dbcp2 -->
<dependency>
  <groupId>org.apache.commons</groupId>
  <artifactId>commons-dbcp2</artifactId>
  <version>2.7.0</version>
</dependency>
```

11. pom.xml에 붙여 넣고 Maven Install 하기

[INFO] BUILD SUCCESS

12. Bean Configuration XML 작성

1)src/config > right-click > New > Spring Bean Configuration File

2)File name : beans.xml

3)Finish

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd">
```

```

117     </beans>
118
119 4)Namespace tab에서 context - http://www.springframework.org/schema/context check
120     -다음 코드 추가
121
122     <context:property-placeholder location="classpath:dbinfo.properties" />
123     <bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource"
124         destroy-method="close">
125         <property name="driverClassName" value="${db.driverClass}" />
126         <property name="url" value="${db.url}" />
127         <property name="username" value="${db.username}" />
128         <property name="password" value="${db.password}" />
129     </bean>
130
131 13. src/com.example.UserClient.java 코드 추가
132
133     package com.example;
134
135     import java.sql.SQLException;
136
137     import javax.sql.DataSource;
138
139     import org.springframework.context.ApplicationContext;
140     import org.springframework.context.support.GenericXmlApplicationContext;
141
142     public class UserClient {
143         public static void main(String[] args) {
144             ApplicationContext ctx = new GenericXmlApplicationContext("classpath:beans.xml");
145
146             DataSource ds = (DataSource) ctx.getBean("dataSource");
147             try{
148                 System.out.println(ds.getConnection());
149             }catch(SQLException ex){
150                 System.out.println(ex);
151             }
152         }
153     }
154
155 14. UserClient.java 실행
156     1932536213, URL=jdbc:oracle:thin:@localhost:1521:XE, UserName=HR, Oracle JDBC
157     driver
158
159 15. Spring JDBC 사용하기
160 1)Maven Repository 에서 'Spring jdbc'로 검색하여 dependency 추가하고 설치
161
162     <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
163     <dependency>
164         <groupId>org.springframework</groupId>
165         <artifactId>spring-jdbc</artifactId>
166         <version>5.2.5.RELEASE</version>
167     </dependency>
168
169 2)pom.xml에 붙여 넣고 Maven Install 하기
170     [INFO] BUILD SUCCESS
171
172 3)beans.xml 수정하기

```

```

173
174     <bean id="dataSource"
      class="org.springframework.jdbc.datasource.SimpleDriverDataSource"
      destroy-method="close">
175         <property name="driverClass" value="${db.driverClass}" />
176         <property name="url" value="${db.url}" />
177         <property name="username" value="${db.username}" />
178         <property name="password" value="${db.password}" />
179     </bean>
180
181 4)UserClient class 실행
182     oracle.jdbc.driver.T4CConnection@2e3967ea
183
184
185 16. c3P0 DataSource 사용하기
186 1)Maven Repository 에서 'c3p0'로 검색하여 dependency 추가하고 설치
187 2)C3P0 click
188 3)0.9.5.5 click
189 4)dependency 복사하여 pom.xml에 붙여넣기
190
191     <!-- https://mvnrepository.com/artifact/com.mchange/c3p0 -->
192     <dependency>
193         <groupId>com.mchange</groupId>
194         <artifactId>c3p0</artifactId>
195         <version>0.9.5.5</version>
196     </dependency>
197
198 5)pom.xml에 붙여 넣고 Maven Install 하기
199     [INFO] BUILD SUCCESS
200
201 6)beans.xml 수정하기
202
203     <bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDataSource"
      destroy-method="close">
204         <property name="driverClass" value="${db.driverClass}" />
205         <property name="jdbcUrl" value="${db.url}" />
206         <property name="user" value="${db.username}" />
207         <property name="password" value="${db.password}" />
208     </bean>
209
210 7)UserClient class 실행
211     com.mchange.v2.c3p0.impl.NewProxyConnection@169bb4dd [wrapping:
      oracle.jdbc.driver.T4CConnection@3fff3a61]
212
213
214 17. JNDI를 이용한 DataSource 사용하기
215 1)beans.xml 수정하기
216     -Namespaces tab에서 jee check
217
218     <?xml version="1.0" encoding="UTF-8"?>
219     <beans xmlns="http://www.springframework.org/schema/beans"
220         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
221         xmlns:context="http://www.springframework.org/schema/context"
222         xmlns:jee="http://www.springframework.org/schema/jee"
223         xsi:schemaLocation="http://www.springframework.org/schema/jee
      http://www.springframework.org/schema/jee/spring-jee-4.3.xsd
      http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans.xsd
224

```

```

225      http://www.springframework.org/schema/context
226      http://www.springframework.org/schema/context/spring-context-4.3.xsd">
227      <context:property-placeholder location="classpath:dbinfo.properties" />
228
229      <jee:jndi-lookup id="dataSource" jndi-name="jdbc/myoracle" resource-ref="true" />
230      <!-- <jee:jndi-lookup> tag를 사용하지 않고 다음과 같은 방법도 가능하다. -->
231      <!-- <bean id="dataSource"
232      class="org.springframework.jndi.JndiObjectFactoryBean">
233      <property name="jndiName" value="jdbc/myoracle" />
234      <property name="resourceRef" value="true" />
235      </bean> -->
236
237      <bean id="dataSource"
238      class="org.springframework.jdbc.datasource.DriverManagerDataSource">
239      <property name="driverClassName" value="${db.driverClass}" />
240      <property name="url" value="${db.url}" />
241      <property name="username" value="${db.username}" />
242      <property name="password" value="${db.password}" />
243      </bean>
244      </beans>

```

2)UserClient class 실행

[oracle.jdbc.driver.T4CConnection@1c7696c6](#)

Task2. Membership Project

1. Table 설계

```

253 CREATE TABLE Users
254 (
255     userid  VARCHAR2(12) NOT NULL PRIMARY KEY,
256     name    VARCHAR2(20) NOT NULL,
257     gender  VARCHAR2(10),
258     city    VARCHAR2(30)
259 );
260
261 INSERT INTO Users VALUES('jimin', '한지민', '여', '서울');
262 COMMIT;

```

2. In Package Explorer > right-click > New > Java Project

- 266 1)Project name : Membership
- 267 2)JRE : Use default JRE 'jdk-13.0.2' and workspace compiler preferences
- 268 3)Next
- 269 4)Uncheck [Create module-info.java file]
- 270 5)Finish

3. vo package 생성

- 274 1)src > right-click > New > Package
- 275 2)Name : com.example.vo
- 276 3)Finish
- 277 4)com.example.vo.UserVO class 생성

```

278
279 package com.example.vo;

```

```
280
281     public class UserVO {
282         private String userId;
283         private String name;
284         private String gender;
285         private String city;
286     }
287
288 4. service package 생성
289 1)src > right-click > New > Package
290 2)Name : com.example.service
291 3)UserService interface 생성
292   -com.example.service > right-click > New > Interface
293   -Name : UserService
294   -Finish
295
296     package com.example.service;
297
298     import java.util.List;
299
300     import com.example.vo.UserVO;
301
302     public interface UserService {
303         void insertUser(UserVO user);
304         List<UserVO> getUserList();
305         void deleteUser(String id);
306         UserVO getUser(String id);
307         void updateUser(UserVO user);
308     }
309
310 4)UserServiceImpl class 생성
311   -com.example.service > right-click > New > Class
312   -Name: UserServiceImpl
313   -Interfaces : com.example.service.UserService
314
315     package com.example.service;
316
317     import java.util.List;
318
319     import com.example.vo.UserVO;
320
321     public class UserServiceImpl implements UserService {
322
323         @Override
324         public void insertUser(UserVO user) {
325             // TODO Auto-generated method stub
326
327         }
328
329         @Override
330         public List<UserVO> getUserList() {
331             // TODO Auto-generated method stub
332             return null;
333         }
334
335         @Override
336         public void deleteUser(String id) {
337             // TODO Auto-generated method stub
```

```
338
339     }
340
341     @Override
342     public UserVO getUser(String id) {
343         // TODO Auto-generated method stub
344         return null;
345     }
346
347     @Override
348     public void updateUser(UserVO user) {
349         // TODO Auto-generated method stub
350
351     }
352
353 }
```

354

355 5. dao package 생성

356 1)src > right-click > New > Package

357 2)Name : com.example.dao

358 3)Finish

359

360 4)UserDao interface 생성

361 -com.example.dao > right-click > New > Interface

362 -Name : UserDao

363 -Finish

364

365 package com.example.dao;

366

367 import java.util.List;

368

369 import com.example.vo.UserVO;

370

```
371
372 public interface UserDao {
373     void insert(UserVO user);
374     List<UserVO> readAll();
375     void update(UserVO user);
376     void delete(String id);
377     UserVO read(String id);
378 }
```

379

380 5)UserDaoImplJDBC class 생성

381 -com.example.dao > right-click > New > Class

382 -Name : UserDaoImplJDBC

383 -Interfaces : com.example.dao.UserDao

384 -Finish

385

386 package com.example.dao;

387

388 import java.util.List;

389

390 import com.example.vo.UserVO;

391

392 public class UserDaoImplJDBC implements UserDao {

393

394 @Override

395 public void insert(UserVO user) {

```

396         // TODO Auto-generated method stub
397
398     }
399
400     @Override
401     public List<UserVO> readAll() {
402         // TODO Auto-generated method stub
403         return null;
404     }
405
406     @Override
407     public void update(UserVO user) {
408         // TODO Auto-generated method stub
409
410     }
411
412     @Override
413     public void delete(String id) {
414         // TODO Auto-generated method stub
415
416     }
417
418     @Override
419     public UserVO read(String id) {
420         // TODO Auto-generated method stub
421         return null;
422     }
423
424 }

```

427 6. Java Project를 Spring Project로 변환

428 1)Membership Project > right-click > Configure > Convert to Maven Project

```

429 -Project : /Membership
430 -Group Id : Membership
431 -Artifact Id : Membership
432 -version : 0.0.1-SNAPSHOT
433 -Packaging : jar
434 -Finish

```

436 2)Membership Project > right-click > Spring > Add Spring Project Nature

439 7. pom.xml 파일에 Spring Context Dependency 추가하기

```

440 <version>0.0.1-SNAPSHOT</version>
441 <dependencies>
442     <dependency>
443         <groupId>org.springframework</groupId>
444         <artifactId>spring-context</artifactId>
445         <version>5.2.5.RELEASE</version>
446     </dependency>
447 </dependencies>

```

449 -pom.xml > right-click > Run As > Maven install

450 [INFO] BUILD SUCCESS 확인

453 8. Oracle Jdbc Driver 설치


```
454 1)Oracle 12C 인 경우
455     <dependency>
456         <groupId>com.oracle</groupId>
457         <artifactId>ojdbc8</artifactId>
458         <version>12.2</version>
459     </dependency>
460
461 2)Oracle 11g 인 경우
462     <dependency>
463         <groupId>com.oracle</groupId>
464         <artifactId>ojdbc6</artifactId>
465         <version>11.2</version>
466     </dependency>
467
468 <참고>
469 -MySQL일 경우에는 'spring mysql'로 검색하여 MySQL Connector/J를 설치한다.
470 <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
471     <dependency>
472         <groupId>mysql</groupId>
473         <artifactId>mysql-connector-java</artifactId>
474         <version>8.0.18</version>
475     </dependency>
476
477
478 9. Spring JDBC 설치
479 1)JdbcTemplate를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.
480
481     <dependency>
482         <groupId>org.springframework</groupId>
483         <artifactId>spring-jdbc</artifactId>
484         <version>5.2.5.RELEASE</version>
485     </dependency>
486
487 2)pom.xml에 붙여 넣고 Maven Install 하기
488 [INFO] BUILD SUCCESS 확인
489
490
491 10. Lombok library 추가
492 1)https://mvnrepository.com/에서 'lombok'으로 검색
493 2)'Project Lombok' click
494 3)1.18.12 click
495 4)dependency copy해서 pom.xml에 붙여넣기
496
497     <dependency>
498         <groupId>org.projectlombok</groupId>
499         <artifactId>lombok</artifactId>
500         <version>1.18.12</version>
501         <scope>provided</scope>
502     </dependency>
503
504 5)pom.xml > right-click > Run As > Maven install
505 [INFO] BUILD SUCCESS 확인
506
507
508 11. resource folder 생성
509 1)Membership project > right-click > New > Source Folder
510 2)Folder name : resources
511 3)Finish
```

```

512
513
514 12. dbinfo.properties 파일 생성
515 1)/resources > right-click > New > File
516 2)File name : dbinfo.properties
517 3)Finish
518
519 db.driverClass=oracle.jdbc.driver.OracleDriver
520 db.url=jdbc:oracle:thin:@localhost:1521:XE
521 db.username=hr
522 db.password=hr
523
524 <참고>
525 3)MySQL일 경우에는 다음과 같이 설정한다.
526 db.driverClass=com.mysql.jdbc.Driver
527 db.url=jdbc:mysql://192.168.136.5:3306/world
528 db.username=root
529 db.password=javamysql
530
531
532 13. Bean Configuration XML 작성
533 1)/resources > right-click > New > Spring Bean Configuration File
534 2)File name : beans.xml
535 3)Finish
536 4)Namespace Tab click
537 5)Check context
538
539 <?xml version="1.0" encoding="UTF-8"?>
540 <beans xmlns="http://www.springframework.org/schema/beans"
541 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
542 xmlns:context="http://www.springframework.org/schema/context"
543 xsi:schemaLocation="http://www.springframework.org/schema/beans
544 http://www.springframework.org/schema/beans/spring-beans.xsd
545 http://www.springframework.org/schema/context
546 http://www.springframework.org/schema/context/spring-context-4.3.xsd">
547
548 <context:property-placeholder location="classpath:dbinfo.properties" />
549 <bean id="dataSource"
550 class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
551 <property name="driverClass" value="${db.driverClass}" />
552 <property name="url" value="${db.url}" />
553 <property name="username" value="${db.username}" />
554 <property name="password" value="${db.password}" />
555 </bean>
556 </beans>
557
558 14. Membership Project의 Bean 등록 및 의존 관계 설정
559 1)<context:component-scan> tag 사용
560 2)@Service, @Repository annotation을 선언한 class들과 @Autowired annotation을 선언하여 의존관
561 계를 설정한 class들이 위치한 package를 Scan하기 위한 설정을 XML에 해주어야 한다.
562 3)beans.xml에 다음 code를 추가한다.
563
564 <context:component-scan base-package="com.example" />
565
566 15. Spring TestContext Framework 사용하기
567 1)MVN Repository에서 'spring test'로 검색하여 'Spring TextContext Framework'을 pom.xml에 추가

```

```

566 <!-- https://mvnrepository.com/artifact/org.springframework/spring-test -->
567 <dependency>
568     <groupId>org.springframework</groupId>
569     <artifactId>spring-test</artifactId>
570     <version>5.2.5.RELEASE</version>
571     <scope>test</scope>
572 </dependency>
573
574 2)MVN Repository에서 'junit로 검색하여 'JUnit Jupiter API'를 선택하여 5.6.2를 pom.xml에 추가
575 <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
576 <dependency>
577     <groupId>org.junit.jupiter</groupId>
578     <artifactId>junit-jupiter-api</artifactId>
579     <version>5.6.2</version>
580     <scope>test</scope>
581 </dependency>
582
583 3)pom.xml Maven Install 하기
584 [INFO] BUILD SUCCESS 확인
585
586 4)src > right-click > New > Package
587 5)Name : com.example.test
588 6)Finish
589 7)com.example.test > right-click > New > JUnit Test Case
590 8)Select [New JUnit Jupiter test]
591 9)Name : MembershipTest
592 10)Finish
593
594 package com.example.test;
595
596 import static org.junit.jupiter.api.Assertions.*;
597
598 import org.junit.jupiter.api.Test;
599 import org.junit.jupiter.api.extension.ExtendWith;
600 import org.springframework.beans.factory.annotation.Autowired;
601 import org.springframework.test.context.ContextConfiguration;
602 import org.springframework.test.context.junit.jupiter.SpringExtension;
603
604 import com.example.service.UserService;
605
606 @ExtendWith(SpringExtension.class)
607 @ContextConfiguration(locations="classpath:beans.xml")
608 class MembershipTest {
609     @Autowired
610     UserService service;
611
612     @Test
613     void test() {
614
615     }
616
617 }
618
619 11)만일 해당 객체를 찾을 수 없다는 오류가 계속 발생하면
620 -해당 Project > right-click > Build Path > Libraries tab
621 -spring-test-5.2.5.RELEASE.jar 선택 후 [Remove] 로 삭제
622 -[Add External JARs...] Click
623 -Local M2 Repository(e.g

```

C:\Users\bluee\.m2\repository\org\springframework\spring-test\5.2.5.RELEASE)에서 직접 jar를 선택할 것

-[Order and Export] tab에서 spring-test-5.2.5.RELEASE.jar 선택 후 [Up] button을 클릭
-해당 Project/src 바로 아래까지 올리고 [Apply and Close] Click

16. UserVO에 lombok Annotation 붙이기

1)com.example.vo.UserVO

```
package com.example.vo;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;
import lombok.ToString;

@Getter
@Setter
@NoArgsConstructor
@AllArgsConstructor
@ToString
public class UserVO {
    private String userId;
    private String name;
    private String gender;
    private String city;
}
```

17. JDBC를 이용한 Membership Project - 사용자 조회 test

1)com.example.dao.UserDaoImplJDBC.java 수정

```
package com.example.dao;

import java.sql.Connection;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.SQLException;
import java.util.List;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.stereotype.Repository;

import com.example.vo.UserVO;

@Repository("userDao")
public class UserDaoImplJDBC implements UserDao {
    @Autowired
    private DataSource dataSource;

    ...

    @Override
    public UserVO read(String id) {
        Connection conn = null;
```

```

680     PreparedStatement pstmt = null;
681     ResultSet rs = null;
682     UserVO userVO = null;
683     try {
684         conn = this.dataSource.getConnection();
685         pstmt = conn.prepareStatement("SELECT * FROM users WHERE userid = ?");
686         pstmt.setString(1, id);
687         rs = pstmt.executeQuery();
688         rs.next();
689         userVO = new UserVO(rs.getString("userid"), rs.getString("name"),
        rs.getString("gender"), rs.getString("city"));
690     } catch (SQLException ex) {
691         System.out.println(ex);
692     } finally {
693         try {
694             if(conn != null) conn.close();
695             if(pstmt != null) pstmt.close();
696             if(rs != null) rs.close();
697         } catch (SQLException ex) {
698             System.out.println(ex);
699         }
700     }
701     return userVO;
702 }
703
704

```

2) com.example.service.UserServiceImpl.java 수정

```

706
707     package com.example.service;
708
709     import java.util.List;
710
711     import org.springframework.beans.factory.annotation.Autowired;
712     import org.springframework.stereotype.Service;
713
714     import com.example.dao.UserDao;
715     import com.example.vo.UserVO;
716
717     @Service("userService")
718     public class UserServiceImpl implements UserService {
719         @Autowired
720         private UserDao userDao;
721         ...
722         ...
723         @Override
724         public UserVO getUser(String id) {
725             return this.userDao.read(id);
726         }
727
728

```

3) com.example.test.MembershipTest.java

```

730
731     @Test
732     public void test() {
733         //사용자 조회 test
734         UserVO user = service.getUser("jimin");
735         System.out.println(user);
736         assertEquals("한지민", user.getName());

```

```

737     }
738
739 4)right-click > Run As > Junit Test
740 5)결과 -> Junit View에 초록색 bar
741     UserVO(userId=jimin, name=한지민, gender=여, city=서울)
742
743
744 18. JDBC를 이용한 Membership Project - 사용자 등록 및 목록 조회 test
745 1)com.example.dao.UserDaoImplJDBC.java code 수정
746     @Override
747     public void insert(UserVO user) {
748         Connection conn = null;
749         PreparedStatement pstmt = null;
750         try {
751             conn = this.dataSource.getConnection();
752             String sql = "INSERT INTO users (userid, name, gender,city) VALUES (?, ?, ?, ?)";
753             pstmt = conn.prepareStatement(sql);
754             pstmt.setString(1, user.getUserId());
755             pstmt.setString(2, user.getName());
756             pstmt.setString(3, user.getGender());
757             pstmt.setString(4, user.getCity());
758             pstmt.executeUpdate();
759             System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" +
760                 user.getName());
761         }catch(SQLException ex) {
762             System.out.println(ex);
763         }finally {
764             try {
765                 if(conn != null) conn.close();
766                 if(pstmt != null) pstmt.close();
767             }catch(SQLException ex) {
768                 System.out.println(ex);
769             }
770         }
771
772     @Override
773     public List<UserVO> readAll() {
774         Connection conn = null;
775         Statement stmt = null;
776         ResultSet rs = null;
777         List<UserVO> userList = null;
778         try {
779             conn = this.dataSource.getConnection();
780             stmt = conn.createStatement();
781             rs = stmt.executeQuery("SELECT * FROM users");
782             userList = new ArrayList<UserVO>();
783             while(rs.next()) {
784                 UserVO userVO = new UserVO(rs.getString("userid"), rs.getString("name"),
785                     rs.getString("gender"), rs.getString("city"));
786                 userList.add(userVO);
787             }
788         }catch(SQLException ex) {
789             System.out.println(ex);
790         }finally {
791             try {
792                 if(conn != null) conn.close();
793                 if(stmt != null) stmt.close();

```

```

793         if(rs != null) rs.close();
794     }catch(SQLException ex) {
795         System.out.println(ex);
796     }
797 }
798 return userList;
799 }

```

2)com.example.service.UserServiceImpl.java code 수정

```

804     @Override
805     public void insertUser(UserVO user) {
806         userDao.insert(user);
807     }
808
809     @Override
810     public List<UserVO> getUserList() {
811         return userDao.readAll();
812     }
813
814

```

3)com.example.test.MembershipTest.java

```

817     ...
818     @Test
819     public void test1() {
820         //사용자 등록 및 목록조회 test
821         this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
822         for(UserVO user : this.service.getUserList()){
823             System.out.println(user);
824         }
825     }
826

```

4)right-click > Run As > Junit Test

5)결과 -> Junit View에 초록색 bar

```

829     UserVO [userId=jimin, name=한지민, gender=여, city=서울]
830     등록된 Record UserId=dooly Name=둘리
831     UserVO [userId=dooly, name=둘리, gender=남, city=경기]
832     UserVO [userId=jimin, name=한지민, gender=여, city=서울]
833
834

```

19. JDBC를 이용한 Membership Project - 사용자 정보 수정 test

1)com.example.dao.UserDaoImplJDBC.java code 수정

```

838     @Override
839     public void update(UserVO user) {
840         Connection conn = null;
841         PreparedStatement pstmt = null;
842         try {
843             conn = this.dataSource.getConnection();
844             String sql = "UPDATE users SET name = ?, gender = ?, city = ? WHERE userid = ?";
845             pstmt = conn.prepareStatement(sql);
846             pstmt.setString(1, user.getName());
847             pstmt.setString(2, user.getGender());
848             pstmt.setString(3, user.getCity());
849             pstmt.setString(4, user.getUserId());
850             pstmt.executeUpdate();

```

```

851         System.out.println("갱신된 Record with ID = " + user.getUserId() );
852     }catch(SQLException ex) {
853         System.out.println(ex);
854     }finally {
855         try {
856             if(conn != null) conn.close();
857             if(pstmt != null) pstmt.close();
858         }catch(SQLException ex) {
859             System.out.println(ex);
860         }
861     }
862 }
863
864

```

2)com.example.service.UserServiceImpl.java code 수정

```

866
867     @Override
868     public void updateUser(UserVO user) {
869         userDao.update(user);
870     }
871
872

```

3)com.example.test.MembershipTest.java

```

873
874
875     @Disabled @Test
876     public void test1() {
877         //사용자 등록 및 목록조회 test
878         this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
879         for(UserVO user : this.service.getUserList()){
880             System.out.println(user);
881         }
882     }
883
884     @Test
885     public void test2() {
886         //사용자 정보 수정 test
887         service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
888         UserVO user = service.getUser("dooly");
889         System.out.println(user);
890     }
891

```

4)right-click > Run As > Junit Test

5)결과 -> Junit View에 초록색 bar

```

894     UserVO [userId=jimin, name=한지민, gender=여, city=서울]
895     갱신된 Record with ID = dooly
896     UserVO [userId=dooly, name=김둘리, gender=여, city=부산]
897
898

```

20. JDBC를 이용한 Membership Project - 사용자 정보 삭제 test

1)com.example.dao.UserDaoImplJDBC.java code 수정

```

901
902     @Override
903     public void delete(String id) {
904         Connection conn = null;
905         PreparedStatement pstmt = null;
906         try {
907             conn = this.dataSource.getConnection();
908             pstmt = conn.prepareStatement("DELETE FROM users WHERE userid = ?");

```



```

909         pstmt.setString(1, id);
910         pstmt.executeUpdate();
911         System.out.println("삭제된 Record with ID = " + id );
912     }catch(SQLException ex) {
913         System.out.println(ex);
914     }finally {
915         try {
916             if(conn != null) conn.close();
917             if(pstmt != null) pstmt.close();
918         }catch(SQLException ex) {
919             System.out.println(ex);
920         }
921     }
922 }

```

925 2)com.example.service.UserServiceImpl.java code 수정

```

926
927     @Override
928     public void deleteUser(String id) {
929         userDao.delete(id);
930     }
931

```

933 3)com.example.test.MembershipTest.java

```

934
935     @Test
936     public void test() {
937         UserVO user = this.service.getUser("jimin");
938         System.out.println(user);
939         assertEquals("한지민", user.getName());
940     }
941     @Disabled @Test
942     public void test1() {
943         this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
944         for(UserVO user : this.service.getUserList()){
945             System.out.println(user);
946         }
947     }
948
949     @Disabled @Test
950     public void test2() {
951         service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
952         UserVO user = service.getUser("dooly");
953         System.out.println(user);
954     }
955
956     @Test
957     public void test3() {
958         //사용자 정보 삭제 test
959         service.deleteUser("dooly");
960         for(UserVO user : service.getUserList()){
961             System.out.println(user);
962         }
963     }
964

```

966 4)right-click > Run As > Junit Test

```

967 5)결과 -> Junit View에 초록색 bar
968 UserVO [userId=jimin, name=한지민, gender=여, city=서울]
969 삭제된 Record with ID = dooly
970 UserVO [userId=jimin, name=한지민, gender=여, city=서울]
971
972
973
974 -----
975 Task3. JdbcTemplate를 이용한 Membership Project
976 1. resources/beans.xml 수정
977
978 <?xml version="1.0" encoding="UTF-8"?>
979 <beans xmlns="http://www.springframework.org/schema/beans"
980 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
981 xmlns:context="http://www.springframework.org/schema/context"
982 xsi:schemaLocation="http://www.springframework.org/schema/beans
983 http://www.springframework.org/schema/beans/spring-beans.xsd
984 http://www.springframework.org/schema/context
985 http://www.springframework.org/schema/context/spring-context-4.3.xsd">
986
987     <context:property-placeholder location="classpath:dbinfo.properties" />
988     <context:component-scan base-package="com.example" />
989
990     <bean id="dataSource"
991         class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
992         <property name="driverClass" value="${db.driverClass}" />
993         <property name="url" value="${db.url}" />
994         <property name="username" value="${db.username}" />
995         <property name="password" value="${db.password}" />
996     </bean>
997
998     <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
999         <property name="dataSource" ref="dataSource" />
1000     </bean>
1001 </beans>
1002
1003 2. 사용자 조회 test
1004 1)com.example.dao.UserDaoImplJDBC.java 복사 후 붙여넣기
1005 2)이름을 UserDaoImplJdbcTemplate로 변경
1006
1007 package com.example.dao;
1008
1009 import java.sql.ResultSet;
1010 import java.sql.SQLException;
1011 import java.util.List;
1012
1013 import org.springframework.beans.factory.annotation.Autowired;
1014 import org.springframework.dao.EmptyResultDataAccessException;
1015 import org.springframework.jdbc.core.JdbcTemplate;
1016 import org.springframework.jdbc.core.RowMapper;
1017 import org.springframework.stereotype.Repository;
1018
1019 import com.example.vo.UserVO;
1020
1021 @Repository("userDao1") <---변경
1022 public class UserDaoImplJdbcTemplate implements UserDao {
1023     @Autowired

```

```
1023     private JdbcTemplate jdbcTemplate;    <--변경
1024
1025     class UserMapper implements RowMapper<UserVO> {
1026         public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
1027             UserVO user = new UserVO();
1028             user.setUserId(rs.getString("userid"));
1029             user.setName(rs.getString("name"));
1030             user.setGender(rs.getString("gender"));
1031             user.setCity(rs.getString("city"));
1032             return user;
1033         }
1034     }
1035
1036     @Override
1037     public void insert(UserVO user) {
1038
1039     }
1040
1041     @Override
1042     public List<UserVO> readAll() {
1043         return null;
1044     }
1045
1046     @Override
1047     public void update(UserVO user) {
1048
1049     }
1050
1051     @Override
1052     public void delete(String id) {
1053
1054     }
1055
1056     @Override
1057     public UserVO read(String id) {
1058         String SQL = "SELECT * FROM users WHERE userid = ?";
1059         try {
1060             UserVO user = jdbcTemplate.queryForObject(SQL, new Object[] { id }, new
                UserMapper());
1061             return user;
1062         } catch (EmptyResultDataAccessException e) {
1063             return null;
1064         }
1065     }
1066 }
```

3)com.example.service.UserServiceImpl.java 수정

```
1070
1071     package com.example.service;
1072
1073     import java.util.List;
1074
1075     import org.springframework.beans.factory.annotation.Autowired;
1076     import org.springframework.stereotype.Service;
1077
1078     import com.example.dao.UserDao;
1079     import com.example.vo.UserVO;
```

```

1080
1081 @Service("userService")
1082 public class UserServiceImpl implements UserService {
1083     @Autowired
1084     private UserDao userDao1;    <--변경
1085
1086     @Override
1087     public void insertUser(UserVO user) {
1088
1089     }
1090
1091     @Override
1092     public List<UserVO> getUserList() {
1093         return null;
1094     }
1095
1096     @Override
1097     public void deleteUser(String id) {
1098
1099     }
1100
1101     @Override
1102     public UserVO getUser(String id) {
1103         return this.userDao1.read(id);
1104     }
1105
1106     @Override
1107     public void updateUser(UserVO user) {
1108
1109     }
1110
1111 }

```

4)/src/test/java/MembershipTest.java

```

1115
1116 @Test
1117 public void test() {
1118     //사용자 조회 test
1119     UserVO user = service.getUser("jimin");
1120     System.out.println(user);
1121     assertEquals("한지민", user.getName());
1122 }

```

5)결과

UserVO(userId=jimin, name=한지민, gender=여, city=서울)

3. 사용자 등록 및 목록 조회 test

1)com.example.daoUserDaoImplJdbcTemplate.java code 수정

```

1131
1132 @Override
1133 public void insert(UserVO user) {
1134     String SQL = "INSERT INTO users (userid, name, gender,city) VALUES (?, ?, ?, ?)";
1135     jdbcTemplate.update(SQL, user.getUserId(), user.getName(), user.getGender(),
1136         user.getCity());
1137     System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" +

```

```

        user.getName());
1137     }
1138
1139     @Override
1140     public List<UserVO> readAll() {
1141         String SQL = "SELECT * FROM users";
1142         List<UserVO> userList = jdbcTemplate.query(SQL, new UserMapper());
1143         return userList;
1144     }
1145
1146

```

2)com.example.service.UserServiceImpl.java code 수정

```

1149     @Override
1150     public void insertUser(UserVO user) {
1151         this.userDao1.insert(user);
1152     }
1153
1154     @Override
1155     public List<UserVO> getUserList() {
1156         return userDao1.readAll();
1157     }
1158
1159

```

3)/src/test/java/MembershipTest.java

```

1162     @Test
1163     public void test1() {
1164         //사용자 등록 및 목록조회 test
1165         this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1166         for(UserVO user : this.service.getUserList()){
1167             System.out.println(user);
1168         }
1169     }
1170
1171

```

4)결과

```

1173     UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1174     등록된 Record UserId=dooly Name=둘리
1175     UserVO(userId=dooly, name=둘리, gender=남, city=경기)
1176     UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1177
1178

```

4. 사용자 정보 수정 test

1)com.example.dao.UserDaoImplJdbcTemplate.java code 수정

```

1182     @Override
1183     public void update(UserVO user) {
1184         String SQL = "UPDATE users SET name = ?, gender = ?, city = ? WHERE userid = ?";
1185         jdbcTemplate.update(SQL, user.getName(), user.getGender(),
1186             user.getCity(),user.getUserId());
1187         System.out.println("갱신된 Record with ID = " + user.getUserId() );
1188     }
1189
1190

```

2)com.example.service.UserServiceImpl.java code 수정

```

1192     @Override

```

```
1193     public void updateUser(UserVO user) {
1194         userDao1.update(user);
1195     }
1196
1197 3)/src/test/java/MembershipTest.java
1198
1199     @Disabled @Test
1200     public void test1() {
1201         //사용자 등록 및 목록조회 test
1202         this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1203         for(UserVO user : this.service.getUserList()){
1204             System.out.println(user);
1205         }
1206     }
1207
1208     @Test
1209     public void test2() {
1210         //사용자 정보 수정 test
1211         service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1212         UserVO user = service.getUser("dooly");
1213         System.out.println(user);
1214     }
1215
1216
1217 4)결과
1218     UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1219     갱신된 Record with ID = dooly
1220     UserVO(userId=dooly, name=김둘리, gender=여, city=부산)
1221
1222
1223 5. 사용자 정보 삭제 test
1224 1)com.example.dao.UserDaoImplJdbcTemplate.java code 수정
1225
1226     @Override
1227     public void delete(String id) {
1228         String SQL = "DELETE FROM users WHERE userid = ?";
1229         jdbcTemplate.update(SQL, id);
1230         System.out.println("삭제된 Record with ID = " + id );
1231     }
1232
1233
1234 2)com.example.service.UserServiceImpl.java 코드 수정
1235
1236     @Override
1237     public void deleteUser(String id) {
1238         userDao1.delete(id);
1239     }
1240
1241
1242 3)/src/test/java/MembershipTest.java
1243
1244     @Test
1245     public void test3() {
1246         //사용자 정보 삭제 test
1247         service.deleteUser("dooly");
1248         for(UserVO user : service.getUserList()){
1249             System.out.println(user);
1250         }
1251     }
```

```

1251     }
1252
1253
1254 4)결과
1255     UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1256     갱신된 Record with ID = dooly
1257     UserVO(userId=dooly, name=김둘리, gender=여, city=부산)
1258     삭제된 Record with ID = dooly
1259     UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1260
1261
1262
1263 -----
1264 Task4. iBATIS를 이용한 Membership Project
1265 1. 준비
1266     1)mvnrepository(https://mvnrepository.com에서 'ibatis'로 검색
1267     2)Ibatis Sqlmap click
1268     3)2.3.4.726 click
1269     4)dependency 복사해서 pom.xml에 넣기
1270         <dependency>
1271             <groupId>org.apache.ibatis</groupId>
1272             <artifactId>ibatis-sqlmap</artifactId>
1273             <version>2.3.4.726</version>
1274         </dependency>
1275
1276 5)pom.xml에 붙여 넣고 Maven Install 하기
1277     [INFO] BUILD SUCCESS 확인
1278     -혹시 Error 발생하면 Project > right-click > Maven > Update Project > 해당 Project 체크 확인후
1279     > OK
1280     -다시 pom.xml > right-click > Run As > Maven install
1281
1282 6)SqlMapConfig.xml 생성
1283     -src > right-click > New > Other > XML > XML File > Next
1284     -File name : SqlMapConfig.xml
1285     -Finish
1286     -<!DOCTYPE element는 Internet에서 sqlmapconfig.xml로 검색
1287
1288     <?xml version="1.0" encoding="UTF-8"?>
1289     <!DOCTYPE sqlMapConfig
1290         PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"
1291         "http://ibatis.apache.org/dtd/sql-map-config-2.dtd">
1292     <sqlMapConfig>
1293         <properties resource="dbinfo.properties" />
1294         <settings useStatementNamespaces="true"/>
1295         <transactionManager type="JDBC">
1296             <dataSource type="SIMPLE">
1297                 <property name="JDBC.Driver" value="${db.driverClass}"/>
1298                 <property name="JDBC.ConnectionURL" value="${db.url}"/>
1299                 <property name="JDBC.Username" value="${db.username}"/>
1300                 <property name="JDBC.Password" value="${db.password}"/>
1301             </dataSource>
1302         </transactionManager>
1303         <sqlMap resource="com/example/dao/Users.xml"/>
1304     </sqlMapConfig>
1305
1306 7)User.xml 파일 생성
1307     -com.example.dao > right-click > New > Other > XML > XML File > Next
1308     -File name : Users.xml

```

```

1308 -Finish
1309
1310 <?xml version="1.0" encoding="UTF-8"?>
1311 <!DOCTYPE sqlMap
1312     PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"
1313     "http://ibatis.apache.org/dtd/sql-map-2.dtd

```



```

1366     }
1367
1368     @Override
1369     public void delete(String id) {
1370
1371     }
1372
1373     @Override
1374     public UserVO read(String id) {
1375         Reader rd = null;
1376         SqlMapClient smc = null;
1377         UserVO userVO = null;
1378         try {
1379             rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1380             smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1381             userVO = (UserVO)smc.queryForObject("Users.useResultMap", id);
1382         } catch (IOException | SQLException e) {
1383             // TODO Auto-generated catch block
1384             e.printStackTrace();
1385         }
1386         return userVO;
1387     }
1388 }

```

3)com.example.service.UserServiceImpl.java 수정

```

1392
1393 package com.example.service;
1394
1395 import java.util.List;
1396
1397 import org.springframework.beans.factory.annotation.Autowired;
1398 import org.springframework.stereotype.Service;
1399
1400 import com.example.dao.UserDao;
1401 import com.example.vo.UserVO;
1402
1403 @Service("userService")
1404 public class UserServiceImpl implements UserService {
1405     @Autowired
1406     private UserDao userDao2; <--변경
1407
1408     @Override
1409     public void insertUser(UserVO user) {
1410     }
1411
1412     @Override
1413     public List<UserVO> getUserList() {
1414         return null;
1415     }
1416
1417     @Override
1418     public void deleteUser(String id) {
1419     }
1420
1421     @Override
1422     public UserVO getUser(String id) {
1423         return userDao2.read(id);

```

```

1424     }
1425
1426     @Override
1427     public void updateUser(UserVO user) {
1428     }
1429 }
1430
1431
1432 4)/src/test/java/MembershipTest.java
1433
1434 @Test
1435 public void test() {
1436     //사용자 조회 test
1437     UserVO user = service.getUser("jimin");
1438     System.out.println(user);
1439     assertEquals("한지민", user.getName());
1440 }
1441
1442 5)결과
1443     UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1444
1445
1446 3. 사용자 등록 및 목록 조회 test
1447 1)Users.xml
1448     <insert id="insert" parameterClass="userVO">
1449         INSERT INTO USERS(userid, name, gender, city)
1450         VALUES (#userId#, #name#, #gender#, #city#)
1451     </insert>
1452
1453     <select id="getAll" resultClass="userVO">
1454         SELECT * FROM USERS
1455     </select>
1456
1457 2)UserDaoImplBatis.java
1458
1459 @Override
1460 public void insert(UserVO user) {
1461     Reader rd = null;
1462     SqlMapClient smc = null;
1463     UserVO userVO = null;
1464     try {
1465         rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1466         smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1467         smc.insert("Users.insert", user);
1468         System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" +
            user.getName());
1469     } catch (IOException | SQLException e) {
1470         // TODO Auto-generated catch block
1471         e.printStackTrace();
1472     }
1473 }
1474
1475 @Override
1476 public List<UserVO> readAll() {
1477     Reader rd = null;
1478     SqlMapClient smc = null;
1479     List<UserVO> userList = null;
1480     try {

```

```

1481         rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1482         smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1483         userList = (List<UserVO>)smc.queryForList("Users.getAll", null);
1484     } catch (IOException | SQLException e) {
1485         // TODO Auto-generated catch block
1486         e.printStackTrace();
1487     }
1488     return userList;
1489 }
1490
1491 3)UserServiceImpl.java
1492
1493     @Override
1494     public void insertUser(UserVO user) {
1495         this.userDao2.insert(user);
1496     }
1497
1498     @Override
1499     public List<UserVO> getUserList() {
1500         return this.userDao2.readAll();
1501     }
1502
1503 4)MembershipTest.java
1504     @Test
1505     public void test1() {
1506         this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1507         for(UserVO user : this.service.getUserList()){
1508             System.out.println(user);
1509         }
1510     }
1511
1512 5)결과
1513     UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1514     등록된 Record UserId=dooly Name=둘리
1515     UserVO(userId=dooly, name=둘리, gender=남, city=경기)
1516     UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1517
1518
1519 4. 사용자 정보 수정 test
1520 1)Users.xml
1521     <update id="update" parameterClass="userVO">
1522         UPDATE USERS
1523         SET   name = #name#, gender = #gender#, city = #city#
1524         WHERE userId = #userId#
1525     </update>
1526
1527 2)com.example.dao.UserDaoImplBatis.java code 수정
1528     @Override
1529     public void update(UserVO user) {
1530         Reader rd = null;
1531         SqlMapClient smc = null;
1532         UserVO userVO = null;
1533         try {
1534             rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1535             smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1536             smc.update("Users.update", user);
1537             System.out.println("갱신된 Record with ID = " + user.getUserId() );
1538         } catch (IOException | SQLException e) {

```

```

1539         // TODO Auto-generated catch block
1540         e.printStackTrace();
1541     }
1542 }
1543
1544 3) UserServiceImpl.java
1545
1546 @Override
1547 public void updateUser(UserVO user) {
1548     this.userDao2.update(user);
1549 }
1550
1551 4) MembershipTest.java 수정
1552 @Disabled @Test
1553 public void test1() {
1554     this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1555     for(UserVO user : this.service.getUserList()){
1556         System.out.println(user);
1557     }
1558 }
1559
1560 @Test
1561 public void test2() {
1562     service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1563     UserVO user = service.getUser("dooly");
1564     System.out.println(user);
1565 }
1566
1567 5) 결과
1568 UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1569 갱신된 Record with ID = dooly
1570 UserVO(userId=dooly, name=김둘리, gender=여, city=부산)
1571
1572
1573 5. 사용자 정보 삭제 test
1574 1) Users.xml
1575 <delete id="delete" parameterClass="String">
1576     DELETE FROM USERS WHERE userid = #id#
1577 </delete>
1578
1579 2) com.example.dao.UserDaoImplBatis.java code 수정
1580 @Override
1581 public void delete(String id) {
1582     Reader rd = null;
1583     SqlMapClient smc = null;
1584     UserVO userVO = null;
1585     try {
1586         rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1587         smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1588         smc.delete("Users.delete", id);
1589         System.out.println("삭제된 Record with ID = " + id );
1590     } catch (IOException | SQLException e) {
1591         // TODO Auto-generated catch block
1592         e.printStackTrace();
1593     }
1594 }
1595
1596 3) UserServiceImpl.java

```

```

1597
1598     @Override
1599     public void deleteUser(String id) {
1600         this.userDao2.delete(id);
1601     }
1602
1603 4)MembershipTest.java 수정
1604     @Test
1605     public void test() {
1606         UserVO user = this.service.getUser("jimin");
1607         System.out.println(user);
1608         assertEquals("한지민", user.getName());
1609     }
1610     @Disabled @Test
1611     public void test1() {
1612         this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1613         for(UserVO user : this.service.getUserList()){
1614             System.out.println(user);
1615         }
1616     }
1617
1618     @Disabled @Test
1619     public void test2() {
1620         service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1621         UserVO user = service.getUser("dooly");
1622         System.out.println(user);
1623     }
1624
1625     @Test
1626     public void test3() {
1627         //사용자 정보 삭제 test
1628         service.deleteUser("dooly");
1629         for(UserVO user : service.getUserList()){
1630             System.out.println(user);
1631         }
1632     }
1633
1634 5)결과
1635     UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1636     삭제된 Record with ID = dooly
1637     UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1638
1639
1640
1641 -----
1642 Task5. MyBatis를 이용한 Membership Project
1643 1. 준비
1644     1)mvnrepository(https://mvnrepository.com에서 'Mybatis'로 검색
1645     2)MyBatis click
1646     3)3.5.4 click
1647     4)dependency를 복사해서 pom.xml에 붙여넣기
1648         <dependency>
1649             <groupId>org.mybatis</groupId>
1650             <artifactId>mybatis</artifactId>
1651             <version>3.5.4</version>
1652         </dependency>
1653
1654     5)mvnrepository(https://mvnrepository.com에서 'Mybatis spring'로 검색

```

```

1655 6)MyBatis Spring click
1656 7)2.0.4 click
1657 8)dependency를 복사해서 pom.xml에 붙여넣기
1658 <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
1659 <dependency>
1660     <groupId>org.mybatis</groupId>
1661     <artifactId>mybatis-spring</artifactId>
1662     <version>2.0.4</version>
1663 </dependency>
1664
1665 9)pom.xml에 붙여 넣고 Maven Install 하기
1666 [INFO] BUILD SUCCESS 확인
1667 -혹시 Error 발생하면 Project > right-click > Maven > Update Project > 해당 Project 체크 확인후
    > OK
1668 -다시 pom.xml > right-click > Run As > Maven install
1669
1670 10)resources/dbinfo.properties
1671 db.driverClass=oracle.jdbc.driver.OracleDriver
1672 db.url=jdbc:oracle:thin:@localhost:1521:XE
1673 db.username=hr
1674 db.password=hr
1675
1676 11)mybatis-config.xml 생성
1677 -src > right-click > New > Other > XML > XML File > Next
1678 -File name : mybatis-config.xml
1679 -Finish
1680
1681 -https://github.com/mybatis/mybatis-3/releases
1682 -mybatis-3.5.4.zip downloads > Unzip
1683 -mybatis-3.5.4.pdf file 참조
1684
1685 <?xml version="1.0" encoding="UTF-8"?>
1686 <!DOCTYPE configuration
1687     PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
1688     "http://mybatis.org/dtd/mybatis-3-config.dtd">
1689 <configuration>
1690     <properties resource="dbinfo.properties" />
1691     <typeAliases>
1692         <typeAlias type="com.example.vo.UserVO" alias="userVO" />
1693     </typeAliases>
1694     <environments default="development">
1695         <environment id="development">
1696             <transactionManager type="JDBC"/>
1697             <dataSource type="POOLED">
1698                 <property name="driver" value="${db.driverClass}"/>
1699                 <property name="url" value="${db.url}"/>
1700                 <property name="username" value="${db.username}"/>
1701                 <property name="password" value="${db.password}"/>
1702             </dataSource>
1703         </environment>
1704     </environments>
1705     <mappers>
1706         <mapper resource="com/example/dao/mybatis-mapper.xml"/>
1707     </mappers>
1708 </configuration>
1709
1710 12)mybatis-mapper.xml 생성
1711 -com.example.dao > right-click > New > Other > XML > XML File

```

```

1712 -File name : mybatis-mapper.xml
1713 -Finish
1714
1715     <?xml version="1.0" encoding="UTF-8"?>
1716     <!DOCTYPE mapper
1717         PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
1718         "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
1719     <mapper namespace="com.example.vo.UserVO">
1720
1721     </mapper>
1722
1723
1724 2. 사용자 조회 test
1725 1)com.example.dao/mybatis-mapper.xml
1726     <resultMap id="userVOResult" type="userVO">
1727         <result property="userId" column="userid" />
1728         <result property="name" column="name" />
1729         <result property="gender" column="gender" />
1730         <result property="city" column="city" />
1731     </resultMap>
1732     <select id="select" parameterType="String" resultType="userVO"
1733         resultMap="userVOResult">
1734         SELECT * FROM USERS WHERE userid = #{id}
1735     </select>
1736 2)com.example.dao.UserDaoImplMyBatis.java 생성
1737 -UserDaoImplBatis.java를 copy하여 paste
1738 -이름 변경 : UserDaoImplMyBatis
1739 -OK
1740
1741     package com.example.dao;
1742
1743     import java.io.IOException;
1744     import java.io.Reader;
1745     import java.util.List;
1746
1747     import org.apache.ibatis.session.SqlSession;
1748     import org.apache.ibatis.session.SqlSessionFactoryBuilder;
1749     import org.springframework.stereotype.Repository;
1750
1751     import com.example.vo.UserVO;
1752     import com.ibatis.common.resources.Resources;
1753
1754     @Repository("userDao3")
1755     public class UserDaoImplMyBatis implements UserDao {
1756
1757         @Override
1758         public void insert(UserVO user) {
1759         }
1760
1761         @Override
1762         public List<UserVO> readAll() {
1763             return null;
1764         }
1765
1766         @Override
1767         public void update(UserVO user) {
1768         }

```

```
1769
1770     @Override
1771     public void delete(String id) {
1772     }
1773
1774     @Override
1775     public UserVO read(String id) {
1776         Reader rd = null;
1777         SqlSession session = null;
1778         UserVO userVO = null;
1779         try {
1780             rd = Resources.getResourceAsReader("mybatis-config.xml");
1781             session = new SqlSessionFactoryBuilder().build(rd).openSession();
1782             userVO = (UserVO)session.selectOne("select", id);
1783         } catch (IOException e) {
1784             e.printStackTrace();
1785         }
1786         return userVO;
1787     }
1788 }
```

3) UserServiceImpl.java 수정

```
1790
1791
1792     package com.example.service;
1793
1794     import java.util.List;
1795
1796     import org.springframework.beans.factory.annotation.Autowired;
1797     import org.springframework.stereotype.Service;
1798
1799     import com.example.dao.UserDao;
1800     import com.example.vo.UserVO;
1801
1802     @Service("userService")
1803     public class UserServiceImpl implements UserService {
1804         @Autowired
1805         private UserDao userDao3; <---변경
1806
1807         @Override
1808         public void insertUser(UserVO user) {
1809         }
1810
1811         @Override
1812         public List<UserVO> getUserList() {
1813             return null;
1814         }
1815
1816         @Override
1817         public void deleteUser(String id) {
1818         }
1819
1820         @Override
1821         public UserVO getUser(String id) {
1822             return this.userDao3.read(id);
1823         }
1824
1825         @Override
1826         public void updateUser(UserVO user) {
```



```

1827     }
1828 }
1829
1830 4)MembershipTest.java
1831
1832     package com.example.test;
1833
1834     import static org.junit.jupiter.api.Assertions.assertEquals;
1835
1836     import org.junit.jupiter.api.Test;
1837     import org.junit.jupiter.api.extension.ExtendWith;
1838     import org.springframework.beans.factory.annotation.Autowired;
1839     import org.springframework.test.context.ContextConfiguration;
1840     import org.springframework.test.context.junit.jupiter.SpringExtension;
1841
1842     import com.example.service.UserService;
1843     import com.example.vo.UserVO;
1844
1845     @ExtendWith(SpringExtension.class)
1846     @ContextConfiguration(locations = "classpath:beans.xml")
1847     class MembershipTest {
1848         @Autowired
1849         UserService service;
1850
1851         @Test
1852         public void test() {
1853             UserVO user = this.service.getUser("jimin");
1854             System.out.println(user);
1855             assertEquals("한지민", user.getName());
1856         }
1857     }
1858
1859 5)결과
1860     UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1861
1862
1863 3. 사용자 등록 및 목록 조회 test
1864 1)mybatis-mapper.xml
1865
1866     <insert id="insert" parameterType="userVO">
1867         INSERT INTO USERS(userid, name, gender, city)
1868         VALUES ({userId}, {name}, {gender}, {city})
1869     </insert>
1870
1871     <select id="selectAll" resultType="userVO" resultMap="userVOResult">
1872         SELECT * FROM USERS
1873     </select>
1874
1875 2)UserDaoImplMyBatis.java
1876
1877     @Override
1878     public void insert(UserVO user) {
1879         Reader rd = null;
1880         SqlSession session = null;
1881         UserVO userVO = null;
1882         try {
1883             rd = Resources.getResourceAsReader("mybatis-config.xml");
1884             session = new SqlSessionFactoryBuilder().build(rd).openSession();

```

```

1885         session.insert("insert", user);
1886         session.commit();
1887         System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" +
            user.getName());
1888     } catch (IOException e) {
1889         e.printStackTrace();
1890     }
1891 }
1892
1893 @Override
1894 public List<UserVO> readAll() {
1895     Reader rd = null;
1896     SqlSession session = null;
1897     List<UserVO> userList = null;
1898     try {
1899         rd = Resources.getResourceAsReader("mybatis-config.xml");
1900         session = new SqlSessionFactoryBuilder().build(rd).openSession();
1901         userList = session.selectList("selectAll");
1902     } catch (IOException e) {
1903         e.printStackTrace();
1904     }
1905     return userList;
1906 }
1907
1908 3)UserServiceImpl.java
1909
1910 @Override
1911 public void insertUser(UserVO user) {
1912     this.userDao3.insert(user);
1913 }
1914
1915 @Override
1916 public List<UserVO> getUserList() {
1917     return this.userDao3.readAll();
1918 }
1919
1920 4)MembershipTest.java
1921
1922 @Test
1923 public void test1() {
1924     this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1925     for(UserVO user : this.service.getUserList()){
1926         System.out.println(user);
1927     }
1928 }
1929
1930 5)결과
1931 UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1932 등록된 Record UserId=dooly Name=둘리
1933 UserVO(userId=dooly, name=둘리, gender=남, city=경기)
1934 UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1935
1936
1937 4. 사용자 정보 수정 test
1938 1)mybatis-mapper.xml
1939
1940 <update id="update" parameterType="userVO">
1941     UPDATE USERS SET name = #{name}, gender = #{gender}, city = #{city}

```

```
1942     WHERE userid = #{userId}
1943 </update>
1944
1945 2) UserDaoImplMyBatis.java
1946
1947     @Override
1948     public void update(UserVO user) {
1949         Reader rd = null;
1950         SqlSession session = null;
1951         UserVO userVO = null;
1952         try {
1953             rd = Resources.getResourceAsReader("mybatis-config.xml");
1954             session = new SqlSessionFactoryBuilder().build(rd).openSession();
1955             session.update("update", user);
1956             session.commit();
1957             System.out.println("갱신된 Record with ID = " + user.getUserId() );
1958         } catch (IOException e) {
1959             e.printStackTrace();
1960         }
1961     }
1962
1963 3) UserServiceImpl.java
1964
1965     @Override
1966     public void updateUser(UserVO user) {
1967         this.userDao3.update(user);
1968     }
1969
1970 4) MembershipTest.java
1971
1972     @Autowired
1973     UserService service;
1974
1975     @Test
1976     public void test() {
1977         UserVO user = this.service.getUser("jimin");
1978         System.out.println(user);
1979         assertEquals("한지민", user.getName());
1980     }
1981     @Disabled @Test
1982     public void test1() {
1983         this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1984         for(UserVO user : this.service.getUserList()){
1985             System.out.println(user);
1986         }
1987     }
1988
1989     @Test
1990     public void test2() {
1991         service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1992         UserVO user = service.getUser("dooly");
1993         System.out.println(user);
1994     }
1995
1996 5)결과
1997     UserVO(userid=jimin, name=한지민, gender=여, city=서울)
1998     갱신된 Record with ID = dooly
1999     UserVO(userid=dooly, name=김둘리, gender=여, city=부산)
```

```
2000
2001
2002 5. 사용자 정보 삭제 test
2003 1)mybatis-mapper.xml
2004
2005     <delete id="delete" parameterType="String">
2006         DELETE FROM USERS WHERE  userid = #{id}
2007     </delete>
2008
2009 2)UserDaoImplMyBatis.java
2010
2011     @Override
2012     public void delete(String id) {
2013         Reader rd = null;
2014         SqlSession session = null;
2015         UserVO userVO = null;
2016         try {
2017             rd = Resources.getResourceAsReader("mybatis-config.xml");
2018             session = new SqlSessionFactoryBuilder().build(rd).openSession();
2019             session.delete("delete", id);
2020             session.commit();
2021             System.out.println("삭제된 Record with ID = " + id );
2022         } catch (IOException e) {
2023             e.printStackTrace();
2024         }
2025     }
2026
2027 3)UserServiceImpl.java
2028
2029     @Override
2030     public void deleteUser(String id) {
2031         this.userDao3.delete(id);
2032     }
2033
2034 4)MembershipTest.java
2035
2036     @Autowired
2037     UserService service;
2038
2039     @Test
2040     public void test() {
2041         UserVO user = this.service.getUser("jimin");
2042         System.out.println(user);
2043         assertEquals("한지민", user.getName());
2044     }
2045     @Disabled @Test
2046     public void test1() {
2047         this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
2048         for(UserVO user : this.service.getUserList()){
2049             System.out.println(user);
2050         }
2051     }
2052
2053     @Disabled @Test
2054     public void test2() {
2055         service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
2056         UserVO user = service.getUser("dooly");
2057         System.out.println(user);
```

```

2058     }
2059
2060     @Test
2061     public void test3() {
2062         //사용자 정보 삭제 test
2063         service.deleteUser("dooly");
2064         for(UserVO user : service.getUserList()){
2065             System.out.println(user);
2066         }
2067     }
2068
2069     5)결과
2070     UserVO(userId=jimin, name=한지민, gender=여, city=서울)
2071     삭제된 Record with ID = dooly
2072     UserVO(userId=jimin, name=한지민, gender=여, city=서울)
2073
2074
2075
2076 -----
2077 Task6. Example of Spring JdbcTemplate
2078 1. Create Table
2079
2080     CREATE TABLE Employee(
2081         id NUMBER(10),
2082         name VARCHAR2(100),
2083         salary NUMBER(10)
2084     );
2085
2086
2087 2. In Package Explorer > right-click > New > Java Project
2088     1)Project Name : JdbcTemplateDemo
2089     2)JRE
2090         -Select [Use default JRE 'jdk-13.0.2' and workspace compiler preferences]
2091     3)Next
2092     4)Uncheck [Create module-info.java file]
2093     5)Finish
2094
2095
2096 3. src > right-click > New > Package
2097     1)Name : com.example
2098     2)Finish
2099
2100
2101 4. Java Project를 Spring Project로 변환
2102     1)JdbcTemplateDemo Project > right-click > Configure > Convert to Maven Project
2103         -Project : /JdbcTemplateDemo
2104         -Group Id : JdbcTemplateDemo
2105         -Artifact Id : JdbcTemplateDemo
2106         -version : 0.0.1-SNAPSHOT
2107         -Packaging : jar
2108         -Finish
2109
2110     2)JdbcTemplateDemo Project > right-click > Spring > Add Spring Project Nature
2111
2112     3)pom.xml file에 Spring Context Dependency 추가하기
2113         <version>0.0.1-SNAPSHOT</version>
2114         <dependencies>
2115             <dependency>

```

```
2116         <groupId>org.springframework</groupId>
2117         <artifactId>spring-context</artifactId>
2118         <version>5.2.5.RELEASE</version>
2119     </dependency>
2120 </dependencies>
2121
2122 4)pom.xml > right-click > Run As > Maven install
2123 [INFO] BUILD SUCCESS 확인
2124
2125
2126 5. Lombok library 추가
2127 1)https://mvnrepository.com/에서 'lombok'으로 검색
2128 2)'Project Lombok' click
2129 3)1.18.12 click
2130 4)dependency copy해서 pom.xml에 붙여넣기
2131
2132     <dependencies>
2133     <dependency>
2134         <groupId>org.springframework</groupId>
2135         <artifactId>spring-context</artifactId>
2136         <version>5.2.5.RELEASE</version>
2137     </dependency>
2138     <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
2139     <dependency>
2140         <groupId>org.projectlombok</groupId>
2141         <artifactId>lombok</artifactId>
2142         <version>1.18.12</version>
2143         <scope>provided</scope>
2144     </dependency>
2145 </dependencies>
2146
2147 5)pom.xml > right-click > Run As > Maven install
2148 [INFO] BUILD SUCCESS 확인
2149
2150
2151 6. pom.xml에 Oracle Jdbc Driver 설정하기
2152 1)pom.xml에 다음 코드 추가
2153
2154     <dependency>
2155         <groupId>com.oracle</groupId>
2156         <artifactId>ojdbc8</artifactId>
2157         <version>12.2</version>
2158     </dependency>
2159
2160 2)pom.xml > right-click > Run As > Maven install
2161 [INFO] BUILD SUCCESS 확인
2162
2163
2164 7. Spring JDBC pom.xml에 추가하기
2165 1)pom.xml에 다음 코드 추가
2166
2167     <dependency>
2168         <groupId>org.springframework</groupId>
2169         <artifactId>spring-jdbc</artifactId>
2170         <version>5.2.5.RELEASE</version>
2171     </dependency>
2172
2173 2)pom.xml > right-click > Run As > Maven install
```

2174 [INFO] BUILD SUCCESS 확인

2175

2176

2177 8. Employee class 생성

2178 1)com.example > right-click > New > Class

2179 2)Name : Employee

2180 3)Finish

2181

2182 package com.example;

2183

2184 import lombok.AllArgsConstructor;

2185 import lombok.Getter;

2186 import lombok.NoArgsConstructor;

2187 import lombok.Setter;

2188

2189 @Getter

2190 @AllArgsConstructor

2191 @NoArgsConstructor

2192 public class Employee {

2193 @Setter private int id;

2194 private String name;

2195 private float salary;

2196 }

2197

2198

2199 9. EmployeeDao class 생성

2200 1)com.example > right-click > New > Class

2201 2)Name : EmployeeDao

2202 3)Finish

2203

2204 package com.example;

2205

2206 import org.springframework.beans.factory.annotation.Autowired;

2207 import org.springframework.jdbc.core.JdbcTemplate;

2208 import org.springframework.stereotype.Repository;

2209

2210 @Repository

2211 public class EmployeeDao {

2212 @Autowired

2213 private JdbcTemplate jdbcTemplate;

2214

2215 public int saveEmployee(Employee e){

2216 String query="INSERT INTO Employee

VALUES(""+e.getId()+","+""+e.getName()+","+""+e.getSalary()+")";

2217 return jdbcTemplate.update(query);

2218 }

2219 public int updateEmployee(Employee e){

2220 String query="Update Employee SET

name="" + e.getName() + ", salary="" + e.getSalary() + " where id="" + e.getId() + " ";

2221 return jdbcTemplate.update(query);

2222 }

2223 public int deleteEmployee(Employee e){

2224 String query="DELETE FROM Employee where id="" + e.getId() + " ";

2225 return jdbcTemplate.update(query);

2226 }

2227 }

2228

2229

```
2230 10. resources folder 생성하기
2231 1)JdbcTemplateDemo project > right-click > New > Source Folder
2232 2)Folder name : resources
2233 3)Finish
2234
2235
2236 11. resources/dbinfo.properties file 생성
2237
2238 db.driverClass=oracle.jdbc.driver.OracleDriver
2239 db.url=jdbc:oracle:thin:@localhost:1521:XE
2240 db.username=hr
2241 db.password=hr
2242
2243
2244 12. Java Annotation 환경설정 파일 생성
2245 1)com.example > right-click > New > Class
2246 2)Name : ApplicationConfig
2247 3)Finish
2248
2249 package com.example;
2250
2251 import javax.sql.DataSource;
2252
2253 import org.springframework.beans.factory.annotation.Value;
2254 import org.springframework.context.annotation.Bean;
2255 import org.springframework.context.annotation.ComponentScan;
2256 import org.springframework.context.support.PropertySourcesPlaceholderConfigurer;
2257 import org.springframework.core.io.ClassPathResource;
2258 import org.springframework.jdbc.core.JdbcTemplate;
2259 import org.springframework.jdbc.datasource.DriverManagerDataSource;
2260
2261 @ComponentScan(basePackages = "com.example")
2262 public class ApplicationConfig {
2263     @Value("${db.driverClass}")
2264     private String driverClassName;
2265     @Value("${db.url}")
2266     private String url;
2267     @Value("${db.username}")
2268     private String username;
2269     @Value("${db.password}")
2270     private String password;
2271
2272     @Bean
2273     public static PropertySourcesPlaceholderConfigurer properties() {
2274         PropertySourcesPlaceholderConfigurer configurer = new
            PropertySourcesPlaceholderConfigurer();
2275         configurer.setLocation(new ClassPathResource("dbinfo.properties"));
2276         return configurer;
2277     }
2278
2279     @Bean
2280     public DataSource dataSource() {
2281         DriverManagerDataSource ds = new DriverManagerDataSource();
2282         ds.setDriverClassName(this.driverClassName);
2283         ds.setUrl(this.url);
2284         ds.setUsername(this.username);
2285         ds.setPassword(this.password);
2286         return ds;
```



```

2287     }
2288
2289     @Bean
2290     public JdbcTemplate jdbcTemplate() {
2291         JdbcTemplate template = new JdbcTemplate();
2292         template.setDataSource(this.dataSource());
2293         return template;
2294     }
2295 }
2296
2297
2298 13. MainClass Class 생성
2299 1)com.example > right-click > New > Class
2300 2)Name : MainClass
2301 3)Finish
2302
2303     package com.example;
2304
2305     import org.springframework.context.ApplicationContext;
2306     import org.springframework.context.annotation.AnnotationConfigApplicationContext;
2307
2308     public class MainClass {
2309         public static void main(String[] args) {
2310             ApplicationContext ctx=new
2311                 AnnotationConfigApplicationContext(ApplicationConfig.class);
2312
2313             EmployeeDao dao = (EmployeeDao)ctx.getBean("empDao");
2314             int status = dao.saveEmployee(new Employee(102,"Amit",35000));
2315             System.out.println("Insert Status = " + status);
2316
2317             status = dao.updateEmployee(new Employee(102,"Sonoo",15000));
2318             System.out.println("Update Status = " + status);
2319
2320             Employee e=new Employee();
2321             e.setId(102);
2322             status = dao.deleteEmployee(e);
2323             System.out.println("Delete Status = " + status);
2324         }
2325     }
2326
2327 4)결과
2328     Insert Status = 1
2329     Update Status = 1
2330     Delete Status = 1
2331
2332
2333 -----
2334 Task7. Example of PreparedStatement in Spring JdbcTemplate
2335 1. EmployeeDao1 class 생성
2336 1)com.example.EmployeeDao를 복사하여 붙여넣기
2337 2)이름변경 : EmployeeDao1
2338 3)OK
2339
2340     package com.example;
2341     import java.sql.PreparedStatement;
2342     import java.sql.SQLException;
2343

```

```

2344 import org.springframework.dao.DataAccessException;
2345 import org.springframework.jdbc.core.JdbcTemplate;
2346 import org.springframework.jdbc.core.PreparedStatementCallback;
2347
2348 public class EmployeeDao {
2349     private JdbcTemplate jdbcTemplate;
2350
2351     public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
2352         this.jdbcTemplate = jdbcTemplate;
2353     }
2354
2355     public Boolean saveEmployeeByPreparedStatement(final Employee e){
2356         String query="INSERT INTO Employee VALUES(?,?,?)";
2357         return jdbcTemplate.execute(query,new PreparedStatementCallback<Boolean>(){
2358             @Override
2359             public Boolean doInPreparedStatement(PreparedStatement ps)
2360                 throws SQLException, DataAccessException {
2361
2362                 ps.setInt(1,e.getId());
2363                 ps.setString(2,e.getName());
2364                 ps.setFloat(3,e.getSalary());
2365
2366                 return ps.execute();
2367             }
2368         });
2369     }
2370 }
2371 }
2372

```

2. AppConfig1 class 생성

1)com.example.ApplicationConfig.java를 복사하여 붙여넣기

2)이름변경 : AppConfig1

3)OK

```

2377
2378 package com.example;
2379
2380 import javax.sql.DataSource;
2381
2382 import org.springframework.beans.factory.annotation.Value;
2383 import org.springframework.context.annotation.Bean;
2384 import org.springframework.context.annotation.ComponentScan;
2385 import org.springframework.context.support.PropertySourcesPlaceholderConfigurer;
2386 import org.springframework.core.io.ClassPathResource;
2387 import org.springframework.jdbc.core.JdbcTemplate;
2388 import org.springframework.jdbc.datasource.DriverManagerDataSource;
2389
2390 @ComponentScan(basePackages = "com.example")
2391 public class AppConfig1 {
2392     @Value("${db.driverClass}")
2393     private String driverClassName;
2394     @Value("${db.url}")
2395     private String url;
2396     @Value("${db.username}")
2397     private String username;
2398     @Value("${db.password}")
2399     private String password;
2400
2401     @Bean

```

```

2402     public static PropertySourcesPlaceholderConfigurer properties() {
2403         PropertySourcesPlaceholderConfigurer configurer = new
            PropertySourcesPlaceholderConfigurer();
2404         configurer.setLocation(new ClassPathResource("dbinfo.properties"));
2405         return configurer;
2406     }
2407
2408     @Bean
2409     public DataSource dataSource() {
2410         DriverManagerDataSource ds = new DriverManagerDataSource();
2411         ds.setDriverClassName(this.driverClassName);
2412         ds.setUrl(this.url);
2413         ds.setUsername(this.username);
2414         ds.setPassword(this.password);
2415         return ds;
2416     }
2417
2418     @Bean
2419     public JdbcTemplate jdbcTemplate() {
2420         JdbcTemplate template = new JdbcTemplate();
2421         template.setDataSource(this.dataSource());
2422         return template;
2423     }
2424
2425     @Bean
2426     public EmployeeDao1 empDao1() {
2427         EmployeeDao1 empDao1 = new EmployeeDao1();
2428         return empDao1;
2429     }
2430 }
2431
2432
2433 3. MainClass1 class 생성
2434 1)com.example.MainClass 복사하여 붙여넣기
2435 2)이름변경 : MainClass1
2436 3)OK
2437
2438 package com.example;
2439
2440 import org.springframework.context.ApplicationContext;
2441 import org.springframework.context.support.ClassPathXmlApplicationContext;
2442 public class Test {
2443
2444     public static void main(String[] args) {
2445         ApplicationContext ctx=new ClassPathXmlApplicationContext("applicationContext.xml");
2446
2447         EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");
2448         dao.saveEmployeeByPreparedStatement(new Employee(108,"Amit",35000));
2449     }
2450 }
2451
2452
2453 -----
2454 Task8. ResultSetExtractor Example | Fetching Records by Spring JdbcTemplate
2455 1. Employee.java 수정
2456
2457 package com.example;

```

```
2458
2459 import lombok.AllArgsConstructor;
2460 import lombok.Getter;
2461 import lombok.NoArgsConstructor;
2462 import lombok.Setter;
2463 import lombok.ToString;
2464
2465 @Getter
2466 @Setter
2467 @AllArgsConstructor
2468 @NoArgsConstructor
2469 @ToString
2470 public class Employee {
2471     private int id;
2472     private String name;
2473     private float salary;
2474 }
2475
2476
2477 2. EmployeeDao2 class 생성
2478 1)com.example.EmployeeDao1를 복사하여 붙여넣기
2479 2)이름변경 : EmployeeDao2
2480 3)OK
2481
2482 package com.example;
2483
2484 import java.sql.ResultSet;
2485 import java.sql.SQLException;
2486 import java.util.ArrayList;
2487 import java.util.List;
2488
2489 import org.springframework.beans.factory.annotation.Autowired;
2490 import org.springframework.dao.DataAccessException;
2491 import org.springframework.jdbc.core.JdbcTemplate;
2492 import org.springframework.jdbc.core.ResultSetExtractor;
2493 import org.springframework.stereotype.Repository;
2494
2495 @Repository
2496 public class EmployeeDao2 {
2497     @Autowired
2498     private JdbcTemplate jdbcTemplate;
2499
2500     public List<Employee> getAllEmployees(){
2501         return jdbcTemplate.query("SELECT * FROM Employee", new
2502             ResultSetExtractor<List<Employee>>(){
2503             @Override
2504                 public List<Employee> extractData(ResultSet rs) throws SQLException,
2505                     DataAccessException {
2506
2507                     List<Employee> list=new ArrayList<Employee>();
2508                     while(rs.next()){
2509                         Employee e=new Employee();
2510                         e.setId(rs.getInt(1));
2511                         e.setName(rs.getString(2));
2512                         e.setSalary(rs.getInt(3));
2513                         list.add(e);
2514                     }
2515                     return list;
2516                 }
2517             }
2518     }
```

```
2515         }
2516     });
2517 }
2518 }
2519
2520
2521 3. ApplicationConfig2 class 생성
2522 1)com.example.ApplicationConfig1.java를 복사하여 붙여넣기
2523 2)이름변경 : ApplicationConfig2
2524 3)OK
2525
2526     ...
2527     @Bean
2528     public EmployeeDao2 empDao2() {
2529         EmployeeDao2 empDao2 = new EmployeeDao2();
2530         return empDao2;
2531     }
2532 }
2533
2534
2535 4. MainClass2 class 생성
2536 1)MainClass1.java copy하여 붙여넣기
2537 2)이름변경 : MainClass2
2538 3)OK
2539
2540     package com.example;
2541
2542     import org.springframework.context.ApplicationContext;
2543     import org.springframework.context.annotation.AnnotationConfigApplicationContext;
2544
2545     public class MainClass2 {
2546         public static void main(String[] args) {
2547             ApplicationContext ctx=new
2548                 AnnotationConfigApplicationContext(ApplicationConfig2.class);
2549
2550             EmployeeDao2 dao2 = (EmployeeDao2)ctx.getBean("empDao2");
2551             dao2.getAllEmployees().forEach(emp -> System.out.println(emp));
2552         }
2553     }
2554
2555 4)결과
2556     Employee(id=108, name=Amit, salary=35000.0)
2557
2558
2559 -----
2560 Task9. RowMapper Example | Fetching records by Spring JdbcTemplate
2561 1. EmployeeDao3 class 생성
2562 1)com.example.EmployeeDao2를 복사하여 붙여넣기
2563 2)이름변경 : EmployeeDao3
2564 3)OK
2565
2566     package com.example;
2567
2568     import java.sql.ResultSet;
2569     import java.sql.SQLException;
2570     import java.util.List;
2571
```

```

2572 import org.springframework.beans.factory.annotation.Autowired;
2573 import org.springframework.jdbc.core.JdbcTemplate;
2574 import org.springframework.jdbc.core.RowMapper;
2575 import org.springframework.stereotype.Repository;
2576
2577 @Repository
2578 public class EmployeeDao3 {
2579     @Autowired
2580     private JdbcTemplate jdbcTemplate;
2581
2582     public List<Employee> getAllEmployeesRowMapper(){
2583         return jdbcTemplate.query("select * from employee",new RowMapper<Employee>(){
2584
2585             @Override
2586             public Employee mapRow(ResultSet rs, int rownumber) throws SQLException {
2587                 Employee e=new Employee();
2588                 e.setId(rs.getInt(1));
2589                 e.setName(rs.getString(2));
2590                 e.setSalary(rs.getInt(3));
2591                 return e;
2592             }
2593         });
2594     }
2595 }

```

2596 2. AppConfig3 class 생성

2597 1)com.example.ApplicationConfig2.java를 복사하여 붙여넣기

2598 2)이름변경 : AppConfig3

2599 3)OK

```

2600
2601
2602 ...
2603 @Bean
2604 public EmployeeDao3 empDao3() {
2605     EmployeeDao3 empDao3 = new EmployeeDao3();
2606     return empDao3;
2607 }
2608
2609

```

2610 3. MainClass3 class 생성

2611 1)MainClass2.java copy하여 붙여넣기

2612 2)이름변경 : MainClass3

2613 3)OK

```

2614
2615 package com.example;
2616
2617 import org.springframework.context.ApplicationContext;
2618 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
2619
2620 public class MainClass3 {
2621     public static void main(String[] args) {
2622         ApplicationContext ctx=new
2623             AnnotationConfigApplicationContext(ApplicationConfig3.class);
2624
2625         EmployeeDao3 dao3 = (EmployeeDao3)ctx.getBean("empDao3");
2626         dao3.getAllEmployeesRowMapper().forEach(emp -> System.out.println(emp));
2627     }
2628 }

```

```
2628
2629 4)결과
2630     Employee(id=108, name=Amit, salary=35000.0)
2631
2632
2633
2634 -----
2635 Task10. Spring NamedParameterJdbcTemplate Example
2636 1. Create Table
2637
2638     CREATE TABLE Employee(
2639         id NUMBER(10),
2640         name VARCHAR2(100),
2641         salary NUMBER(10)
2642     );
2643
2644
2645 2. In Package Explorer > right-click > New > Java Project
2646     1)Project Name : NamedJdbcTemplateDemo
2647     2)JRE
2648         -Select [Use default JRE 'jdk-13.0.2' and workspace compiler preferences]
2649     3)Next
2650     4)Uncheck [Create module-info.java file]
2651     5)Finish
2652
2653
2654 3. src > right-click > New > Package
2655     1)Name : com.example
2656     2)Finish
2657
2658
2659 4. Java Project를 Spring Project로 변환
2660     1)NamedJdbcTemplateDemo Project > right-click > Configure > Convert to Maven Project
2661         -Project : /NamedJdbcTemplateDemo
2662         -Group Id : NamedJdbcTemplateDemo
2663         -Artifact Id : NamedJdbcTemplateDemo
2664         -version : 0.0.1-SNAPSHOT
2665         -Packaging : jar
2666         -Finish
2667
2668     2)NamedJdbcTemplateDemo Project > right-click > Spring > Add Spring Project Nature
2669
2670     3)pom.xml file에 Spring Context Dependency 추가하기
2671         <version>0.0.1-SNAPSHOT</version>
2672         <dependencies>
2673             <dependency>
2674                 <groupId>org.springframework</groupId>
2675                 <artifactId>spring-context</artifactId>
2676                 <version>5.2.5.RELEASE</version>
2677             </dependency>
2678         </dependencies>
2679
2680     4)pom.xml > right-click > Run As > Maven install
2681         [INFO] BUILD SUCCESS 확인
2682
2683
2684 5. Lombok library 추가
2685     1)<a href="https://mvnrepository.com/">https://mvnrepository.com/</a>에서 'lombok'으로 검색
```

```
2686 2)'Project Lombok' click
2687 3)1.18.12 click
2688 4)dependency copy해서 pom.xml에 붙여넣기
2689
2690     <dependency>
2691         <groupId>org.projectlombok</groupId>
2692         <artifactId>lombok</artifactId>
2693         <version>1.18.12</version>
2694         <scope>provided</scope>
2695     </dependency>
2696
2697 5)pom.xml > right-click > Run As > Maven install
2698 [INFO] BUILD SUCCESS 확인
2699
2700
2701 6. pom.xml에 Oracle Jdbc Driver 설정하기
2702 1)pom.xml에 다음 코드 추가
2703
2704     <dependency>
2705         <groupId>com.oracle</groupId>
2706         <artifactId>ojdbc8</artifactId>
2707         <version>12.2</version>
2708     </dependency>
2709
2710 2)pom.xml > right-click > Run As > Maven install
2711 [INFO] BUILD SUCCESS 확인
2712
2713
2714 7. Spring JDBC pom.xml에 추가하기
2715 1)pom.xml에 다음 코드 추가
2716
2717     <dependency>
2718         <groupId>org.springframework</groupId>
2719         <artifactId>spring-jdbc</artifactId>
2720         <version>5.2.5.RELEASE</version>
2721     </dependency>
2722
2723 2)pom.xml > right-click > Run As > Maven install
2724 [INFO] BUILD SUCCESS 확인
2725
2726
2727 8. Employee class 생성
2728 1)com.example > right-click > New > Class
2729 2)Name : Employee
2730 3)Finish
2731
2732     package com.example;
2733
2734     import lombok.AllArgsConstructor;
2735     import lombok.Getter;
2736
2737     @Getter
2738     @AllArgsConstructor
2739     public class Employee {
2740         private int id;
2741         private String name;
2742         private float salary;
2743     }
```



```
2744
2745
2746 9. EmployeeDao class 생성
2747 1)com.example > New > Class
2748 2)Name : EmployeeDao
2749 3)Finish
2750
2751     package com.example;
2752
2753     import java.sql.PreparedStatement;
2754     import java.sql.SQLException;
2755     import java.util.HashMap;
2756     import java.util.Map;
2757
2758     import org.springframework.beans.factory.annotation.Autowired;
2759     import org.springframework.dao.DataAccessException;
2760     import org.springframework.jdbc.core.PreparedStatementCallback;
2761     import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
2762     import org.springframework.stereotype.Repository;
2763
2764     @Repository
2765     public class EmployeeDao {
2766         @Autowired
2767         private NamedParameterJdbcTemplate template;
2768
2769         public void save (Employee emp){
2770             String query="INSERT INTO Employee VALUES (:id,:name,:salary)";
2771
2772             Map<String,Object> map = new HashMap<String,Object>();
2773             map.put("id", emp.getId());
2774             map.put("name", emp.getName());
2775             map.put("salary", emp.getSalary());
2776
2777             template.execute(query, map, new PreparedStatementCallback<Integer>() {
2778                 @Override
2779                 public Integer doInPreparedStatement(PreparedStatement ps)
2780                     throws SQLException, DataAccessException {
2781                     return ps.executeUpdate();
2782                 }
2783             });
2784         }
2785     }
2786
2787
2788 10. resources folder 생성하기
2789 1)NamedJdbcTemplateDemo project > right-click > New > Source Folder
2790 2)Folder name : resources
2791 3)Finish
2792
2793
2794 11. resources/dbinfo.properties file 생성
2795
2796     db.driverClass=oracle.jdbc.driver.OracleDriver
2797     db.url=jdbc:oracle:thin:@localhost:1521:XE
2798     db.username=hr
2799     db.password=hr
2800
2801
```

```
2802 12. ApplicationConfig class 생성
2803 1)com.example > New > Class
2804 2)Name : ApplicationConfig
2805 3)Finish
2806
2807 package com.example;
2808
2809 import javax.sql.DataSource;
2810
2811 import org.springframework.beans.factory.annotation.Value;
2812 import org.springframework.context.annotation.Bean;
2813 import org.springframework.context.annotation.ComponentScan;
2814 import org.springframework.context.annotation.Configuration;
2815 import org.springframework.context.support.PropertySourcesPlaceholderConfigurer;
2816 import org.springframework.core.io.ClassPathResource;
2817 import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
2818 import org.springframework.jdbc.datasource.DriverManagerDataSource;
2819
2820 @Configuration
2821 @ComponentScan(basePackages = "com.example")
2822 public class ApplicationConfig {
2823     @Value("${db.driverClass}")
2824     private String driverClassName;
2825     @Value("${db.url}")
2826     private String url;
2827     @Value("${db.username}")
2828     private String username;
2829     @Value("${db.password}")
2830     private String password;
2831
2832     @Bean
2833     public static PropertySourcesPlaceholderConfigurer properties() {
2834         PropertySourcesPlaceholderConfigurer configurer = new
            PropertySourcesPlaceholderConfigurer();
2835         configurer.setLocation(new ClassPathResource("dbinfo.properties"));
2836         return configurer;
2837     }
2838
2839     @Bean
2840     public DataSource dataSource() {
2841         DriverManagerDataSource ds = new DriverManagerDataSource();
2842         ds.setDriverClassName(this.driverClassName);
2843         ds.setUrl(this.url);
2844         ds.setUsername(this.username);
2845         ds.setPassword(this.password);
2846         return ds;
2847     }
2848
2849     @Bean
2850     public NamedParameterJdbcTemplate jdbcTemplate() {
2851         NamedParameterJdbcTemplate template = new
            NamedParameterJdbcTemplate(this.dataSource());
2852         return template;
2853     }
2854
2855     @Bean
2856     public EmployeeDao empDao() {
2857         EmployeeDao empDao = new EmployeeDao();
```

```
2858         return empDao;
2859     }
2860 }
2861
2862
2863 13. MainClass class 생성
2864 1)com.example > New > Class
2865 2)Name : MainClass
2866 3)Finish
2867
2868 package com.example;
2869
2870 import org.springframework.beans.factory.BeanFactory;
2871 import org.springframework.beans.factory.xml.XmlBeanFactory;
2872 import org.springframework.core.io.ClassPathResource;
2873 import org.springframework.core.io.Resource;
2874
2875 public class SimpleTest {
2876     public static void main(String[] args) {
2877
2878         Resource r=new ClassPathResource("applicationContext.xml");
2879         BeanFactory factory=new XmlBeanFactory(r);
2880
2881         EmpDao dao=(EmpDao)factory.getBean("edao");
2882         dao.save(new Emp(23,"sonoo",50000));
2883
2884     }
2885 }
```