HOL : Spring JDBC

Task1. Spring Jdbc Demo
1. SpringJdbcDemo project 생성
  1)New > Java Project >
  2)Project name : SpringJdbcDemo > Finish
  3)JRE
    -Select [Use default JRE 'jdk-13.0.2' and workspace compiler preferences]
  4)Next
  5)Uncheck [Create module-info.java file]
  6)Finish

2. com.example Package 생성
  1)src > right-click > New > Package
  2)Name : com.example
  3)Finish

3. config folder 생성
  1)SpringJdbcDemo project > right-click > New > Source Folder
  2)Folder name : config
  3)Finish

4. config/dbinfo.properties file 생성
  1)config > right-click > New > File
  2)File name : dbinfo.properties
  3)Finish

    db.driverClass=oracle.jdbc.driver.OracleDriver
    db.url=jdbc:oracle:thin:@localhost:1521:XE
    db.username=hr
    db.password=hr

5. UserClient.java 생성
  1)src > com.example > right-click > New > Class
  2)Name : UserClient
  3)Finish

    public class UserClient{
      public static void main(String [] args){

      }
    }

6. Maven Project로 전환
  1)SpringJdbcDemo Project > right-click > Configure > Convert to Maven Project
  2)Finish

7. Spring Project로 전환
  1)SpringJdbcDemo Project > right-click > Spring Tools > Add Spring Project Nature

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)depency 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">

```

```
    </beans>

  4)Namespace tab에서 context - http://www.springframework.org/schema/context check
    -다음 코드 추가

    <context:property-placeholder location="classpath:dbinfo.properties" />
    <bean id="dataSource" class="org.apache.commons.dbcp2.BasicDataSource"
    destroy-method="close">
      <property name="driverClassName" value="${db.driverClass}" />
      <property name="url" value="${db.url}" />
      <property name="username" value="${db.username}" />
      <property name="password" value="${db.password}" />
    </bean>


13. src/com.example.UserClient.java 코드 추가

    package com.example;

    import java.sql.SQLException;

    import javax.sql.DataSource;

    import org.springframework.context.ApplicationContext;
    import org.springframework.context.support.GenericXmlApplicationContext;

    public class UserClient {
      public static void main(String[] args) {
        ApplicationContext ctx = new GenericXmlApplicationContext("classpath:beans.xml");

        DataSource ds = (DataSource) ctx.getBean("dataSource");
        try{
          System.out.println(ds.getConnection());
        }catch(SQLException ex){
          System.out.println(ex);
        }
      }
    }

14. UserClient.java 실행
  1932536213, URL=jdbc:oracle:thin:@localhost:1521:XE, UserName=HR, Oracle JDBC
  driver


15. Spring JDBC 사용하기
  1)Maven Repository 에서 'Spring jdbc'로 검색하여 dependency 추가하고 설치

    <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-jdbc</artifactId>
      <version>5.2.5.RELEASE</version>
    </dependency>

  2)pom.xml에 붙여 넣고 Maven Install 하기
    [INFO] BUILD SUCCESS

  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
224           http://www.springframework.org/schema/beans
              http://www.springframework.org/schema/beans/spring-beans.xsd
```

```
            http://www.springframework.org/schema/context
            http://www.springframework.org/schema/context/spring-context-4.3.xsd">

        <context:property-placeholder location="classpath:dbinfo.properties" />

        <jee:jndi-lookup id="dataSource" jndi-name="jdbc/myoracle" resource-ref="true" />
        <!-- <jee:jndi-lookup> tag를 사용하지 않고 다음과 같은 방법도 가능하다. -->
        <!-- <bean id="dataSource"
        class="org.springframework.jndi.JndiObjectFactoryBean">
          <property name="jndiName" value="jdbc/myoracle" />
          <property name="resourceRef" value="true" />
        </bean> -->

        <bean id="dataSource"
        class="org.springframework.jdbc.datasource.DriverManagerDataSource">
          <property name="driverClassName" value="${db.driverClass}" />
          <property name="url" value="${db.url}" />
          <property name="username" value="${db.username}" />
          <property name="password" value="${db.password}" />
        </bean>
      </beans>

  2)UserClient class 실행
    oracle.jdbc.driver.T4CConnection@1c7696c6



-------------------------------------------------
Task2. Membership Project
1. Table 설계

  CREATE TABLE Users
  (
    userid    VARCHAR2(12)  NOT NULL PRIMARY KEY,
    name      VARCHAR2(20)  NOT NULL,
    gender    VARCHAR2(10),
    city      VARCHAR2(30)
  );

  INSERT INTO Users VALUES('jimin', '한지민', '여', '서울');
  COMMIT;


2. In Package Explorer > right-click > New > Java Project
  1)Project name : Membership
  2)JRE : Use default JRE 'jdk-13.0.2' and workspace compiler preferences
  3)Next
  4)Uncheck [Create module-info.java file]
  5)Finish


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

    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
356 5. dao package 생성
357   1)src > right-click > New > Package
358   2)Name : com.example.dao
359   3)Finish
360
361   4)UserDao interface 생성
362     -com.example.dao > right-click > New > Interface
363     -Name : UserDao
364     -Finish
365
366       package com.example.dao;
367
368       import java.util.List;
369
370       import com.example.vo.UserVO;
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       }
425
426
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
435
436   2)Membership Project > right-click > Spring > Add Spring Project Nature
437
438
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>
448
449   -pom.xml > right-click > Run As > Maven install
450     [INFO] BUILD SUCCESS 확인
451
452
453 8. Oracle Jdbc Driver 설치
```

```
  1)Oracle 12C 인 경우
    <dependency>
      <groupId>com.oracle</groupId>
      <artifactId>ojdbc8</artifactId>
      <version>12.2</version>
    </dependency>

  2)Oracle 11g 인 경우
    <dependency>
      <groupId>com.oracle</groupId>
      <artifactId>ojdbc6</artifactId>
      <version>11.2</version>
    </dependency>

  <참고>
  -MySQL일 경우에는 'spring mysql'로 검색하여 MySQL Connector/J를 설치한다.
    <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
    <dependency>
       <groupId>mysql</groupId>
       <artifactId>mysql-connector-java</artifactId>
       <version>8.0.18</version>
    </dependency>


9. Spring JDBC 설치
  1)JdbcTemplate를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.

    <dependency>
      <groupId>org.springframework</groupId>
      <artifactId>spring-jdbc</artifactId>
      <version>5.2.5.RELEASE</version>
    </dependency>

  2)pom.xml에 붙여 넣고 Maven Install 하기
    [INFO] BUILD SUCCESS 확인


10. Lombok library 추가
  1)https://mvnrepository.com/에서 'lombok'으로 검색
  2)'Project Lombok' click
  3)1.18.12 click
  4)depency copy해서 pom.xml에 붙여넣기

    <dependency>
      <groupId>org.projectlombok</groupId>
      <artifactId>lombok</artifactId>
      <version>1.18.12</version>
      <scope>provided</scope>
    </dependency>

  5)pom.xml > right-click > Run As > Maven install
    [INFO] BUILD SUCCESS 확인


11. resource folder 생성
  1)Membership project > right-click > New > Source Folder
  2)Folder name : resources
  3)Finish
```

12. dbinfo.properties 파일 생성
  1)/resources > right-click > New > File
  2)File name : dbinfo.properties
  3)Finish

```
db.driverClass=oracle.jdbc.driver.OracleDriver
db.url=jdbc:oracle:thin:@localhost:1521:XE
db.username=hr
db.password=hr
```

  <참고>
  3)MySQL일 경우에는 다음과 같이 설정한다.

```
db.driverClass=com.mysql.jdbc.Driver
db.url=jdbc:mysql://192.168.136.5:3306/world
db.username=root
db.password=javamysql
```

13. Bean Configuration XML 작성
  1)/resources > right-click > New > Spring Bean Configuration File
  2)File name : beans.xml
  3)Finish
  4)Namespace Tab click
  5)Check context

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns:context="http://www.springframework.org/schema/context"
  xsi:schemaLocation="http://www.springframework.org/schema/beans
  http://www.springframework.org/schema/beans/spring-beans.xsd
    http://www.springframework.org/schema/context
    http://www.springframework.org/schema/context/spring-context-4.3.xsd">

  <context:property-placeholder location="classpath:dbinfo.properties" />
  <bean id="dataSource"
  class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
    <property name="driverClass" value="${db.driverClass}" />
    <property name="url" value="${db.url}" />
    <property name="username" value="${db.username}" />
    <property name="password" value="${db.password}" />
  </bean>
</beans>
```

14. Membership Project의 Bean 등록 및 의존 관계 설정
  1)<context:component-scan> tag 사용
  2)@Service, @Repository annotation을 선언한 class들과 @Autowired annotation을 선언하여 의존관계를 설정한 class들이 위치한 package를 Scan하기 위한 설정을 XML에 해주어야 한다.
  3)beans.xml에 다음 code를 추가한다.

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

15. Spring TestContext Framework 사용하기
  1)MVN Repository에서 'spring test'로 검색하여 'Spring TextContext Framework'을 pom.xml에 추가

```
    <!-- https://mvnrepository.com/artifact/org.springframework/spring-test -->
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-test</artifactId>
        <version>5.2.5.RELEASE</version>
        <scope>test</scope>
    </dependency>

2)MVN Repository에서 'junit로 검색하여 'JUnit Jupiter API'를 선택하여 5.6.2를 pom.xml에 추가
    <!-- https://mvnrepository.com/artifact/org.junit.jupiter/junit-jupiter-api -->
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.6.2</version>
        <scope>test</scope>
    </dependency>

3)pom.xml Maven Install 하기
    [INFO] BUILD SUCCESS 확인

4)src > right-click > New > Package
5)Name : com.example.test
6)Finish
7)com.example.test > right-click > New > JUnit Test Case
8)Select [New JUnit Jupiter test]
9)Name : MembershipTest
10)Finish

    package com.example.test;

    import static org.junit.jupiter.api.Assertions.*;

    import org.junit.jupiter.api.Test;
    import org.junit.jupiter.api.extension.ExtendWith;
    import org.springframework.beans.factory.annotation.Autowired;
    import org.springframework.test.context.ContextConfiguration;
    import org.springframework.test.context.junit.jupiter.SpringExtension;

    import com.example.service.UserService;

    @ExtendWith(SpringExtension.class)
    @ContextConfiguration(locations="classpath:beans.xml")
    class MembershipTest {
        @Autowired
        UserService service;

        @Test
        void test() {

        }

    }

11)만일 해당 객체를 찾을 수 없다는 오류가 계속 발생하면
    -해당 Project > right-click > Build Path > Libraries tab
    -spring-test-5.2.5.RELEASE.jar 선택 후 [Remove] 로 삭제
    -[Add External JARs...] Click
    -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
705   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
729   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=" +
            user.getName());
760       }catch(SQLException ex) {
761         System.out.println(ex);
762       }finally {
763         try {
764           if(conn != null) conn.close();
765           if(pstmt != null) pstmt.close();
766         }catch(SQLException ex) {
767           System.out.println(ex);
768         }
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"),
              rs.getString("gender"), rs.getString("city"));
785           userList.add(userVO);
786         }
787       }catch(SQLException ex) {
788         System.out.println(ex);
789       }finally {
790         try {
791           if(conn != null) conn.close();
792           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       }
800
801
802     2)com.example.service.UserServiceImpl.java code 수정
803
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
815     3)com.example.test.MembershipTest.java
816
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
827     4)right-click > Run As > Junit Test
828     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
835   19. JDBC를 이용한 Membership Project - 사용자 정보 수정 test
836     1)com.example.dao.UserDaoImplJDBC.java code 수정
837
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
865     2)com.example.service.UserServiceImpl.java code 수정
866
867       @Override
868       public void updateUser(UserVO user) {
869         userDao.update(user);
870       }
871
872
873     3)com.example.test.MembershipTest.java
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
892     4)right-click > Run As > Junit Test
893     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
899   20. JDBC를 이용한 Membership Project - 사용자 정보 삭제 test
900     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       }
923
924
925   2)com.example.service.UserServiceImpl.java code 수정
926
927       @Override
928       public void deleteUser(String id) {
929         userDao.delete(id);
930       }
931
932
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
965
966   4)right-click > Run As > Junit Test
```

5)결과 -> Junit View에 초록색 bar
UserVO [userId=jimin, name=한지민, gender=여, city=서울]
삭제된 Record with ID = dooly
UserVO [userId=jimin, name=한지민, gender=여, city=서울]

21. Java File로 환경설정을 SimpleDataSource 구현하기
1)Membership/resources/beans.xml 삭제하기

2)com.example.config package 생성하기

3)com.example.config.ApplicationConfig class 생성하기

```
package com.example.config;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.support.PropertySourcesPlaceholderConfigurer;
import org.springframework.core.io.ClassPathResource;
import org.springframework.jdbc.datasource.DriverManagerDataSource;

@Configuration
@ComponentScan(basePackages = {"com.example"})
public class ApplicationConfig {
  @Value("${db.driverClass}")
  private String driverClass;
  @Value("${db.url}")
  private String url;
  @Value("${db.username}")
  private String username;
  @Value("${db.password}")
  private String password;

  @Bean
  public static PropertySourcesPlaceholderConfigurer properties() {
    PropertySourcesPlaceholderConfigurer configure = new
    PropertySourcesPlaceholderConfigurer();
    configure.setLocation(new ClassPathResource("dbinfo.properties"));
    return configure;
  }
  @Bean
  public DataSource dataSource() {
    DriverManagerDataSource dataSource = new DriverManagerDataSource();
    dataSource.setDriverClassName(this.driverClass);
    dataSource.setUrl(this.url);
    dataSource.setUsername(this.username);
    dataSource.setPassword(this.password);

    return dataSource;
  }
}
```

4)com.example.test package 생성하기
5)com.example.test.MainClass class 생성하기

```
1024    package com.example.test;
1025
1026    import org.springframework.context.ApplicationContext;
1027    import org.springframework.context.annotation.AnnotationConfigApplicationContext;
1028
1029    import com.example.config.ApplicationConfig;
1030    import com.example.service.UserService;
1031    import com.example.vo.UserVO;
1032
1033    public class MainClass {
1034      public static void main(String[] args) {
1035        ApplicationContext ctx = new
          AnnotationConfigApplicationContext(ApplicationConfig.class);
1036        UserService userService = ctx.getBean("userService", UserService.class);
1037        UserVO userVO = userService.getUser("jimin");
1038        System.out.println(userVO);
1039      }
1040    }
1041
1042  6)com.example.test.UserJUnit5SpringTest class 생성하기
1043
1044    package com.example.test;
1045
1046
1047    import static org.junit.jupiter.api.Assertions.assertEquals;
1048
1049    import org.junit.jupiter.api.Disabled;
1050    import org.junit.jupiter.api.Test;
1051    import org.junit.jupiter.api.extension.ExtendWith;
1052    import org.springframework.beans.factory.annotation.Autowired;
1053    import org.springframework.test.context.ContextConfiguration;
1054    import org.springframework.test.context.junit.jupiter.SpringExtension;
1055
1056    import com.example.config.ApplicationConfig;
1057    import com.example.service.UserService;
1058    import com.example.vo.UserVO;
1059
1060
1061    @ExtendWith(SpringExtension.class)
1062    @ContextConfiguration(classes = { ApplicationConfig.class })
1063    public class UserJUnit5SpringTest {
1064      @Autowired
1065      private UserService userService;
1066
1067      @Disabled @Test
1068      public void test() {
1069        UserVO userVO = this.userService.getUser("jimin");
1070        System.out.println(userVO);
1071      }
1072
1073      @Disabled @Test
1074      public void test1() {
1075        // 사용자 등록 및 목록조회 test
1076        this.userService.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1077      }
1078
1079      @Disabled @Test
1080      public void test2() {
```

```
1081          assertEquals(2, this.userService.getUserList().size());
1082          this.userService.getUserList().forEach(user -> System.out.println(user));
1083        }
1084
1085        @Disabled @Test
1086        public void test3() {
1087          // 사용자 정보 수정 test
1088          this.userService.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1089          UserVO user = this.userService.getUser("dooly");
1090          assertEquals("김둘리", user.getName());
1091          System.out.println(user);
1092        }
1093
1094        @Test
1095        public void test4() {
1096          // 사용자 정보 삭제 test
1097          this.userService.deleteUser("dooly");
1098          this.userService.getUserList().forEach(user -> System.out.println(user));
1099        }
1100      }
1101
1102
1103  ------------------------------------------------------------
1104  Task3. JdbcTemplate를 이용한 Membership Project
1105  1. resources/beans.xml 수정
1106
1107    <?xml version="1.0" encoding="UTF-8"?>
1108    <beans xmlns="http://www.springframework.org/schema/beans"
1109      xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1110      xmlns:context="http://www.springframework.org/schema/context"
1111      xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd
1112          http://www.springframework.org/schema/context
              http://www.springframework.org/schema/context/spring-context-4.3.xsd">
1113
1114      <context:property-placeholder location="classpath:dbinfo.properties" />
1115      <context:component-scan base-package="com.example" />
1116
1117      <bean id="dataSource"
1118        class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
1119        <property name="driverClass" value="${db.driverClass}" />
1120        <property name="url" value="${db.url}" />
1121        <property name="username" value="${db.username}" />
1122        <property name="password" value="${db.password}" />
1123      </bean>
1124
1125      <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
1126        <property name="dataSource" ref="dataSource" />
1127      </bean>
1128    </beans>
1129
1130
1131  2. 사용자 조회 test
1132    1)com.example.dao.UserDaoImplJDBC.java 복사 후 붙여넣기
1133    2)이름을 UserDaoImplJdbcTemplate로 변경
1134
1135      package com.example.dao;
1136
```

```
1137    import java.sql.ResultSet;
1138    import java.sql.SQLException;
1139    import java.util.List;
1140
1141    import org.springframework.beans.factory.annotation.Autowired;
1142    import org.springframework.dao.EmptyResultDataAccessException;
1143    import org.springframework.jdbc.core.JdbcTemplate;
1144    import org.springframework.jdbc.core.RowMapper;
1145    import org.springframework.stereotype.Repository;
1146
1147    import com.example.vo.UserVO;
1148
1149    @Repository("userDao1")  <---변경
1150    public class UserDaoImplJdbcTemplate implements UserDao {
1151      @Autowired
1152      private JdbcTemplate jdbcTemplate;    <--변경
1153
1154      class UserMapper implements RowMapper<UserVO> {
1155        public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
1156          UserVO user = new UserVO();
1157          user.setUserId(rs.getString("userid"));
1158          user.setName(rs.getString("name"));
1159          user.setGender(rs.getString("gender"));
1160          user.setCity(rs.getString("city"));
1161          return user;
1162        }
1163      }
1164
1165      @Override
1166      public void insert(UserVO user) {
1167
1168      }
1169
1170      @Override
1171      public List<UserVO> readAll() {
1172        return null;
1173      }
1174
1175      @Override
1176      public void update(UserVO user) {
1177
1178      }
1179
1180      @Override
1181      public void delete(String id) {
1182
1183      }
1184
1185      @Override
1186      public UserVO read(String id) {
1187        String SQL = "SELECT * FROM users WHERE userid = ?";
1188        try {
1189          UserVO user = jdbcTemplate.queryForObject(SQL, new Object[] { id }, new
              UserMapper());
1190          return user;
1191        } catch (EmptyResultDataAccessException e) {
1192          return null;
1193        }
```

```
1194        }
1195      }
1196
1197
1198    3)com.example.service.UserServiceImpl.java 수정
1199
1200      package com.example.service;
1201
1202      import java.util.List;
1203
1204      import org.springframework.beans.factory.annotation.Autowired;
1205      import org.springframework.stereotype.Service;
1206
1207      import com.example.dao.UserDao;
1208      import com.example.vo.UserVO;
1209
1210      @Service("userService")
1211      public class UserServiceImpl implements UserService {
1212        @Autowired
1213        private UserDao userDao1;    <--변경
1214
1215        @Override
1216        public void insertUser(UserVO user) {
1217
1218        }
1219
1220        @Override
1221        public List<UserVO> getUserList() {
1222          return null;
1223        }
1224
1225        @Override
1226        public void deleteUser(String id) {
1227
1228        }
1229
1230        @Override
1231        public UserVO getUser(String id) {
1232          return this.userDao1.read(id);
1233        }
1234
1235        @Override
1236        public void updateUser(UserVO user) {
1237
1238        }
1239
1240      }
1241
1242
1243    4)/src/test/java/MembershipTest.java
1244
1245      @Test
1246      public void test() {
1247        //사용자 조회 test
1248        UserVO user = service.getUser("jimin");
1249        System.out.println(user);
1250        assertEquals("한지민", user.getName());
1251      }
```

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

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

@Override
public List<UserVO> readAll() {
  String SQL = "SELECT * FROM users";
  List<UserVO>  userList = jdbcTemplate.query(SQL, new UserMapper());
  return userList;
}
```

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

```
@Override
public void insertUser(UserVO user) {
  this.userDao1.insert(user);
}

@Override
public List<UserVO> getUserList() {
  return userDao1.readAll();
}
```

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

```
@Test
public void test1() {
  //사용자 등록 및 목록조회 test
  this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
  for(UserVO user : this.service.getUserList()){
    System.out.println(user);
  }
}
```

4)결과
UserVO(userId=jimin, name=한지민, gender=여, city=서울)
등록된 Record UserId=dooly Name=둘리
UserVO(userId=dooly, name=둘리, gender=남, city=경기)
UserVO(userId=jimin, name=한지민, gender=여, city=서울)

4. 사용자 정보 수정 test
1)com.example.dao.UserDaoImplJdbcTemplate.java code 수정

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

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

```
@Override
public void updateUser(UserVO user) {
  userDao1.update(user);
}
```

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

```
@Disabled @Test
public void test1() {
  //사용자 등록 및 목록조회 test
  this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
  for(UserVO user : this.service.getUserList()){
    System.out.println(user);
  }
}

@Test
public void test2() {
  //사용자 정보 수정 test
  service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
  UserVO user = service.getUser("dooly");
  System.out.println(user);
}
```

4)결과

```
UserVO(userId=jimin, name=한지민, gender=여, city=서울)
갱신된 Record with ID = dooly
UserVO(userId=dooly, name=김둘리, gender=여, city=부산)
```

5. 사용자 정보 삭제 test
1)com.example.dao.UserDaoImplJdbcTemplate.java code 수정

```
@Override
public void delete(String id) {
  String SQL = "DELETE FROM users WHERE userid = ?";
  jdbcTemplate.update(SQL, id);
  System.out.println("삭제된 Record with ID = " + id );
}
```

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

```
    @Override
    public void deleteUser(String id) {
      userDao1.delete(id);
    }


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

    @Test
    public void test3() {
      //사용자 정보 삭제 test
      service.deleteUser("dooly");
      for(UserVO user : service.getUserList()){
        System.out.println(user);
      }
    }


 4)결과
    UserVO(userId=jimin, name=한지민, gender=여, city=서울)
    갱신된 Record with ID = dooly
    UserVO(userId=dooly, name=김둘리, gender=여, city=부산)
    삭제된 Record with ID = dooly
    UserVO(userId=jimin, name=한지민, gender=여, city=서울)



------------------------------------------------------------
Task4. iBATIS를 이용한 Membership Project
1. 준비
 1)mvnrepository(https://mvnrepository.com에서 'ibatis'로 검색
 2)Ibatis Sqlmap click
 3)2.3.4.726 click
 4)dependency 복사해서 pom.xml에 넣기
    <dependency>
      <groupId>org.apache.ibatis</groupId>
      <artifactId>ibatis-sqlmap</artifactId>
      <version>2.3.4.726</version>
    </dependency>

 5)pom.xml에 붙여 넣고 Maven Install 하기
    [INFO] BUILD SUCCESS 확인
    -혹시 Error 발생하면 Project > right-click > Maven > Update Project > 해당 Project 체크 확인후
    > OK
    -다시 pom.xml > right-click > Run As > Maven install

 6)SqlMapConfig.xml 생성
    -src > right-click > New > Other > XML > XML File > Next
    -File name : SqlMapConfig.xml
    -Finish
    -<!DOCTYPE element는 Internet에서 sqlmapconfig.xml로 검색

      <?xml version="1.0" encoding="UTF-8"?>
      <!DOCTYPE sqlMapConfig
          PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"
            "http://ibatis.apache.org/dtd/sql-map-config-2.dtd">
      <sqlMapConfig>
        <properties resource="dbinfo.properties" />
```

```
        <settings useStatementNamespaces="true"/>
        <transactionManager type="JDBC">
          <dataSource type="SIMPLE">
            <property name="JDBC.Driver" value="${db.driverClass}"/>
            <property name="JDBC.ConnectionURL" value="${db.url}"/>
            <property name="JDBC.Username" value="${db.username}"/>
            <property name="JDBC.Password" value="${db.password}"/>
          </dataSource>
        </transactionManager>
        <sqlMap resource="com/example/dao/Users.xml"/>
      </sqlMapConfig>

  7)User.xml 파일 생성
    -com.example.dao > right-click > New > Other > XML > XML File > Next
    -File name : Users.xml
    -Finish

      <?xml version="1.0" encoding="UTF-8"?>
      <!DOCTYPE sqlMap
            PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"
              "http://ibatis.apache.org/dtd/sql-map-2.dtd">
      <sqlMap namespace="Users">
        <typeAlias alias="userVO" type="com.example.vo.UserVO"/>
      </sqlMap>


2. 사용자 조회 Test
  1)Users.xml
    <resultMap id="result" class="userVO">
      <result property="userId" column="userid"/>
      <result property="name" column="name"/>
      <result property="gender" column="gender"/>
      <result property="city" column="city"/>
    </resultMap>
    <select id="useResultMap" resultMap="result">
      SELECT * FROM users WHERE userid=#id#
    </select>


  2)com.example.dao.UserDaoImpliBatis.java 생성
    -UserDaoImplJdbcTemplate.java 복사하여 붙여넣기
    -이름변경 : UserDaoImpliBatis

      package com.example.dao;

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

      import org.springframework.stereotype.Repository;

      import com.example.vo.UserVO;
      import com.ibatis.common.resources.Resources;
      import com.ibatis.sqlmap.client.SqlMapClient;
      import com.ibatis.sqlmap.client.SqlMapClientBuilder;

      @Repository("userDao2")  <--변경
```

```
1480        public class UserDaoImpliBatis implements UserDao {
1481
1482          @Override
1483          public void insert(UserVO user) {
1484
1485          }
1486
1487          @Override
1488          public List<UserVO> readAll() {
1489            return null;
1490          }
1491
1492          @Override
1493          public void update(UserVO user) {
1494
1495          }
1496
1497          @Override
1498          public void delete(String id) {
1499
1500          }
1501
1502          @Override
1503          public UserVO read(String id) {
1504            Reader rd = null;
1505            SqlMapClient smc = null;
1506            UserVO userVO = null;
1507            try {
1508              rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1509              smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1510              userVO = (UserVO)smc.queryForObject("Users.useResultMap", id);
1511            } catch (IOException | SQLException e) {
1512              // TODO Auto-generated catch block
1513              e.printStackTrace();
1514            }
1515            return userVO;
1516          }
1517        }
1518
1519
1520    3)com.example.service.UserServiceImpl.java 수정
1521
1522      package com.example.service;
1523
1524        import java.util.List;
1525
1526        import org.springframework.beans.factory.annotation.Autowired;
1527        import org.springframework.stereotype.Service;
1528
1529        import com.example.dao.UserDao;
1530        import com.example.vo.UserVO;
1531
1532        @Service("userService")
1533        public class UserServiceImpl implements UserService {
1534          @Autowired
1535          private UserDao userDao2;   <--변경
1536
1537          @Override
```

```
1538          public void insertUser(UserVO user) {
1539          }
1540
1541          @Override
1542          public List<UserVO> getUserList() {
1543            return null;
1544          }
1545
1546          @Override
1547          public void deleteUser(String id) {
1548          }
1549
1550          @Override
1551          public UserVO getUser(String id) {
1552            return userDao2.read(id);
1553          }
1554
1555          @Override
1556          public void updateUser(UserVO user) {
1557          }
1558        }
1559
1560
1561    4)/src/test/java/MembershipTest.java
1562
1563      @Test
1564      public void test() {
1565        //사용자 조회 test
1566        UserVO user = service.getUser("jimin");
1567        System.out.println(user);
1568        assertEquals("한지민", user.getName());
1569      }
1570
1571    5)결과
1572      UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1573
1574
1575  3. 사용자 등록 및 목록 조회 test
1576    1)Users.xml
1577      <insert id="insert" parameterClass="userVO">
1578        INSERT INTO USERS(userid, name, gender, city)
1579        VALUES (#userId#, #name#, #gender#, #city#)
1580      </insert>
1581
1582      <select id="getAll" resultClass="userVO">
1583        SELECT * FROM USERS
1584      </select>
1585
1586    2)UserDaoImpliBatis.java
1587
1588      @Override
1589      public void insert(UserVO user) {
1590        Reader rd = null;
1591        SqlMapClient smc = null;
1592        UserVO userVO = null;
1593        try {
1594          rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1595          smc = SqlMapClientBuilder.buildSqlMapClient(rd);
```

```
1596            smc.insert("Users.insert", user);
1597            System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" +
                user.getName());
1598          } catch (IOException | SQLException e) {
1599            // TODO Auto-generated catch block
1600            e.printStackTrace();
1601          }
1602        }
1603
1604        @Override
1605        public List<UserVO> readAll() {
1606          Reader rd = null;
1607          SqlMapClient smc = null;
1608          List<UserVO> userList = null;
1609          try {
1610            rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1611            smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1612            userList = (List<UserVO>)smc.queryForList("Users.getAll", null);
1613          } catch (IOException | SQLException e) {
1614            // TODO Auto-generated catch block
1615            e.printStackTrace();
1616          }
1617          return userList;
1618        }
1619
1620      3)UserServiceImpl.java
1621
1622        @Override
1623        public void insertUser(UserVO user) {
1624          this.userDao2.insert(user);
1625        }
1626
1627        @Override
1628        public List<UserVO> getUserList() {
1629          return this.userDao2.readAll();
1630        }
1631
1632      4)MembershipTest.java
1633        @Test
1634        public void test1() {
1635          this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1636          for(UserVO user : this.service.getUserList()){
1637            System.out.println(user);
1638          }
1639        }
1640
1641      5)결과
1642        UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1643        등록된 Record UserId=dooly Name=둘리
1644        UserVO(userId=dooly, name=둘리, gender=남, city=경기)
1645        UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1646
1647
1648  4. 사용자 정보 수정 test
1649      1)Users.xml
1650        <update id="update" parameterClass="userVO">
1651          UPDATE USERS
1652          SET    name = #name#, gender = #gender#, city = #city#
```

```
      WHERE  userId = #userId#
    </update>

  2)com.example.dao.UserDaoImpliBatis.java code 수정
    @Override
    public void update(UserVO user) {
      Reader rd = null;
      SqlMapClient smc = null;
      UserVO userVO = null;
      try {
        rd = Resources.getResourceAsReader("SqlMapConfig.xml");
        smc = SqlMapClientBuilder.buildSqlMapClient(rd);
        smc.update("Users.update", user);
        System.out.println("갱신된 Record with ID = " + user.getUserId() );
      } catch (IOException | SQLException e) {
        // TODO Auto-generated catch block
        e.printStackTrace();
      }
    }

  3)UserServiceImpl.java

    @Override
    public void updateUser(UserVO user) {
      this.userDao2.update(user);
    }

  4)MembershipTest.java 수정
    @Disabled @Test
    public void test1() {
      this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
      for(UserVO user : this.service.getUserList()){
        System.out.println(user);
      }
    }

    @Test
    public void test2() {
      service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
      UserVO user = service.getUser("dooly");
      System.out.println(user);
    }

  5)결과
    UserVO(userId=jimin, name=한지민, gender=여, city=서울)
    갱신된 Record with ID = dooly
    UserVO(userId=dooly, name=김둘리, gender=여, city=부산)


5. 사용자 정보 삭제 test
  1)Users.xml
    <delete id="delete" parameterClass="String">
      DELETE FROM USERS WHERE  userid = #id#
    </delete>

  2)com.example.dao.UserDaoImpliBatis.java code 수정
    @Override
    public void delete(String id) {
```

```
        Reader rd = null;
        SqlMapClient smc = null;
        UserVO userVO = null;
        try {
          rd = Resources.getResourceAsReader("SqlMapConfig.xml");
          smc = SqlMapClientBuilder.buildSqlMapClient(rd);
          smc.delete("Users.delete", id);
          System.out.println("삭제된 Record with ID = " + id );
        } catch (IOException | SQLException e) {
          // TODO Auto-generated catch block
          e.printStackTrace();
        }
      }

  3)UserServiceImpl.java

      @Override
      public void deleteUser(String id) {
        this.userDao2.delete(id);
      }

  4)MembershipTest.java 수정
      @Test
      public void test() {
        UserVO user = this.service.getUser("jimin");
        System.out.println(user);
        assertEquals("한지민", user.getName());
      }
      @Disabled @Test
      public void test1() {
        this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
        for(UserVO user : this.service.getUserList()){
          System.out.println(user);
        }
      }

      @Disabled @Test
      public void test2() {
        service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
        UserVO user = service.getUser("dooly");
        System.out.println(user);
      }

      @Test
      public void test3() {
        //사용자 정보 삭제 test
        service.deleteUser("dooly");
        for(UserVO user : service.getUserList()){
          System.out.println(user);
        }
      }

  5)결과
      UserVO(userId=jimin, name=한지민, gender=여, city=서울)
      삭제된 Record with ID = dooly
      UserVO(userId=jimin, name=한지민, gender=여, city=서울)
```

```
1769
1770 ----------------------------------------------------------------
1771 Task5. MyBatis를 이용한 Membership Project
1772 1. 준비
1773   1)mvnrepository(https://mvnrepository.com에서 'Mybatis'로 검색
1774   2)MyBatis click
1775   3)3.5.4 click
1776   4)dependency를 복사해서 pom.xml에 붙여넣기
1777     <dependency>
1778       <groupId>org.mybatis</groupId>
1779       <artifactId>mybatis</artifactId>
1780       <version>3.5.4</version>
1781     </dependency>
1782
1783   5)mvnrepository(https://mvnrepository.com에서 'Mybatis spring'로 검색
1784   6)MyBatis Spring click
1785   7)2.0.4 click
1786   8)dependency를 복사해서 pom.xml에 붙여넣기
1787     <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
1788     <dependency>
1789        <groupId>org.mybatis</groupId>
1790        <artifactId>mybatis-spring</artifactId>
1791        <version>2.0.4</version>
1792     </dependency>
1793
1794   9)pom.xml에 붙여 넣고 Maven Install 하기
1795     [INFO] BUILD SUCCESS 확인
1796     -혹시 Error 발생하면 Project > right-click > Maven > Update Project > 해당 Project 체크 확인후
       > OK
1797     -다시 pom.xml > right-click > Run As > Maven install
1798
1799   10)resources/dbinfo.properties
1800     db.driverClass=oracle.jdbc.driver.OracleDriver
1801     db.url=jdbc:oracle:thin:@localhost:1521:XE
1802     db.username=hr
1803     db.password=hr
1804
1805   11)mybatis-config.xml 생성
1806     -src > right-click > New > Other > XML > XML File > Next
1807     -File name : mybatis-config.xml
1808     -Finish
1809
1810     -https://github.com/mybatis/mybatis-3/releases
1811     -mybatis-3.5.4.zip downloads > Unzip
1812     -mybatis-3.5.4.pdf file 참조
1813
1814       <?xml version="1.0" encoding="UTF-8"?>
1815       <!DOCTYPE configuration
1816               PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
1817                 "http://mybatis.org/dtd/mybatis-3-config.dtd">
1818       <configuration>
1819         <properties resource="dbinfo.properties" />
1820         <typeAliases>
1821           <typeAlias type="com.example.vo.UserVO" alias="userVO" />
1822         </typeAliases>
1823         <environments default="development">
1824           <environment id="development">
1825             <transactionManager type="JDBC"/>
```

```
            <dataSource type="POOLED">
              <property name="driver" value="${db.driverClass}"/>
              <property name="url" value="${db.url}"/>
              <property name="username" value="${db.username}"/>
              <property name="password" value="${db.password}"/>
            </dataSource>
          </environment>
        </environments>
        <mappers>
          <mapper resource="com/example/dao/mybatis-mapper.xml"/>
        </mappers>
      </configuration>

  12)mybatis-mapper.xml 생성
    -com.example.dao > right-click > New > Other > XML > XML File
    -File name : mybatis-mapper.xml
    -Finish

      <?xml version="1.0" encoding="UTF-8"?>
      <!DOCTYPE mapper
            PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
              "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
      <mapper namespace="com.example.vo.UserVO">

      </mapper>


2. 사용자 조회 test
  1)com.example.dao/mybatis-mapper.xml
    <resultMap id="userVOResult" type="userVO">
      <result property="userId" column="userid" />
      <result property="name" column="name" />
      <result property="gender" column="gender" />
      <result property="city" column="city" />
    </resultMap>
    <select id="select" parameterType="String" resultType="userVO"
    resultMap="userVOResult">
      SELECT * FROM USERS WHERE userid = #{id}
    </select>

  2)com.example.dao.UserDaoImplMyBatis.java 생성
    -UserDaoImpliBatis.java를 copy하여 paste
    -이름변경 : UserDaoImplMyBatis
    -OK

      package com.example.dao;

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

      import org.apache.ibatis.session.SqlSession;
      import org.apache.ibatis.session.SqlSessionFactoryBuilder;
      import org.springframework.stereotype.Repository;

      import com.example.vo.UserVO;
      import com.ibatis.common.resources.Resources;

```

```
@Repository("userDao3")
public class UserDaoImplMyBatis implements UserDao {

  @Override
  public void insert(UserVO user) {
  }

  @Override
  public List<UserVO> readAll() {
    return null;
  }

  @Override
  public void update(UserVO user) {
  }

  @Override
  public void delete(String id) {
  }

  @Override
  public UserVO read(String id) {
    Reader rd = null;
    SqlSession session = null;
    UserVO userVO = null;
    try {
      rd = Resources.getResourceAsReader("mybatis-config.xml");
      session = new SqlSessionFactoryBuilder().build(rd).openSession();
      userVO = (UserVO)session.selectOne("select", id);
    } catch (IOException e) {
      e.printStackTrace();
    }
    return userVO;
  }
}
```

3)UserServiceImpl.java 수정

```
package com.example.service;

import java.util.List;

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

import com.example.dao.UserDao;
import com.example.vo.UserVO;

@Service("userService")
public class UserServiceImpl implements UserService {
  @Autowired
  private UserDao userDao3;   <---변경

  @Override
  public void insertUser(UserVO user) {
  }

  @Override
```

```
1941        public List<UserVO> getUserList() {
1942          return null;
1943        }
1944
1945        @Override
1946        public void deleteUser(String id) {
1947        }
1948
1949        @Override
1950        public UserVO getUser(String id) {
1951          return this.userDao3.read(id);
1952        }
1953
1954        @Override
1955        public void updateUser(UserVO user) {
1956        }
1957      }
1958
1959    4)MembershipTest.java
1960
1961      package com.example.test;
1962
1963      import static org.junit.jupiter.api.Assertions.assertEquals;
1964
1965      import org.junit.jupiter.api.Test;
1966      import org.junit.jupiter.api.extension.ExtendWith;
1967      import org.springframework.beans.factory.annotation.Autowired;
1968      import org.springframework.test.context.ContextConfiguration;
1969      import org.springframework.test.context.junit.jupiter.SpringExtension;
1970
1971      import com.example.service.UserService;
1972      import com.example.vo.UserVO;
1973
1974      @ExtendWith(SpringExtension.class)
1975      @ContextConfiguration(locations = "classpath:beans.xml")
1976      class MembershipTest {
1977        @Autowired
1978        UserService service;
1979
1980        @Test
1981        public void test() {
1982          UserVO user = this.service.getUser("jimin");
1983          System.out.println(user);
1984          assertEquals("한지민", user.getName());
1985        }
1986      }
1987
1988    5)결과
1989      UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1990
1991
1992  3. 사용자 등록 및 목록 조회 test
1993    1)mybatis-mapper.xml
1994
1995      <insert id="insert" parameterType="userVO">
1996        INSERT INTO USERS(userid, name, gender, city)
1997        VALUES (#{userId}, #{name}, #{gender}, #{city})
1998      </insert>
```

```
1999
2000      <select id="selectAll" resultType="userVO" resultMap="userVOResult">
2001        SELECT * FROM USERS
2002      </select>
2003
2004    2)UserDaoImplMyBatis.java
2005
2006      @Override
2007      public void insert(UserVO user) {
2008        Reader rd = null;
2009        SqlSession session = null;
2010        UserVO userVO = null;
2011        try {
2012          rd = Resources.getResourceAsReader("mybatis-config.xml");
2013          session = new SqlSessionFactoryBuilder().build(rd).openSession();
2014          session.insert("insert", user);
2015          session.commit();
2016          System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" +
              user.getName());
2017        } catch (IOException e) {
2018          e.printStackTrace();
2019        }
2020      }
2021
2022      @Override
2023      public List<UserVO> readAll() {
2024        Reader rd = null;
2025        SqlSession session = null;
2026        List<UserVO> userList = null;
2027        try {
2028          rd = Resources.getResourceAsReader("mybatis-config.xml");
2029          session = new SqlSessionFactoryBuilder().build(rd).openSession();
2030          userList = session.selectList("selectAll");
2031        } catch (IOException e) {
2032          e.printStackTrace();
2033        }
2034        return userList;
2035      }
2036
2037    3)UserServiceImpl.java
2038
2039      @Override
2040      public void insertUser(UserVO user) {
2041        this.userDao3.insert(user);
2042      }
2043
2044      @Override
2045      public List<UserVO> getUserList() {
2046        return this.userDao3.readAll();
2047      }
2048
2049    4)MembershipTest.java
2050
2051      @Test
2052      public void test1() {
2053        this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
2054        for(UserVO user : this.service.getUserList()){
2055          System.out.println(user);
```

```
      }
    }

  5)결과
    UserVO(userId=jimin, name=한지민, gender=여, city=서울)
    등록된 Record UserId=dooly Name=둘리
    UserVO(userId=dooly, name=둘리, gender=남, city=경기)
    UserVO(userId=jimin, name=한지민, gender=여, city=서울)


4. 사용자 정보 수정 test
  1)mybatis-mapper.xml

    <update id="update" parameterType="userVO">
      UPDATE USERS SET name = #{name}, gender = #{gender}, city = #{city}
      WHERE userid = #{userId}
    </update>

  2)UserDaoImplMyBatis.java

    @Override
    public void update(UserVO user) {
      Reader rd = null;
      SqlSession session = null;
      UserVO userVO = null;
      try {
        rd = Resources.getResourceAsReader("mybatis-config.xml");
        session = new SqlSessionFactoryBuilder().build(rd).openSession();
        session.update("update", user);
        session.commit();
        System.out.println("갱신된 Record with ID = " + user.getUserId() );
      } catch (IOException e) {
        e.printStackTrace();
      }
    }

  3)UserServiceImpl.java

    @Override
    public void updateUser(UserVO user) {
      this.userDao3.update(user);
    }

  4)MembershipTest.java

    @Autowired
    UserService service;

    @Test
    public void test() {
      UserVO user = this.service.getUser("jimin");
      System.out.println(user);
      assertEquals("한지민", user.getName());
    }
    @Disabled @Test
    public void test1() {
      this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
      for(UserVO user : this.service.getUserList()){
```

```
2114          System.out.println(user);
2115        }
2116      }
2117
2118      @Test
2119      public void test2() {
2120        service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
2121        UserVO user = service.getUser("dooly");
2122        System.out.println(user);
2123      }
2124
2125    5)결과
2126      UserVO(userId=jimin, name=한지민, gender=여, city=서울)
2127      갱신된 Record with ID = dooly
2128      UserVO(userId=dooly, name=김둘리, gender=여, city=부산)
2129
2130
2131  5. 사용자 정보 삭제 test
2132    1)mybatis-mapper.xml
2133
2134        <delete id="delete" parameterType="String">
2135          DELETE FROM USERS WHERE  userid = #{id}
2136        </delete>
2137
2138    2)UserDaoImplMyBatis.java
2139
2140      @Override
2141      public void delete(String id) {
2142        Reader rd = null;
2143        SqlSession session = null;
2144        UserVO userVO = null;
2145        try {
2146          rd = Resources.getResourceAsReader("mybatis-config.xml");
2147          session = new SqlSessionFactoryBuilder().build(rd).openSession();
2148          session.delete("delete", id);
2149          session.commit();
2150          System.out.println("삭제된 Record with ID = " + id );
2151        } catch (IOException e) {
2152          e.printStackTrace();
2153        }
2154      }
2155
2156    3)UserServiceImpl.java
2157
2158      @Override
2159      public void deleteUser(String id) {
2160        this.userDao3.delete(id);
2161      }
2162
2163    4)MembershipTest.java
2164
2165      @Autowired
2166      UserService service;
2167
2168      @Test
2169      public void test() {
2170        UserVO user = this.service.getUser("jimin");
2171        System.out.println(user);
```

```
2172        assertEquals("한지민", user.getName());
2173      }
2174      @Disabled @Test
2175      public void test1() {
2176        this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
2177        for(UserVO user : this.service.getUserList()){
2178          System.out.println(user);
2179        }
2180      }
2181
2182      @Disabled @Test
2183      public void test2() {
2184        service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
2185        UserVO user = service.getUser("dooly");
2186        System.out.println(user);
2187      }
2188
2189      @Test
2190      public void test3() {
2191        //사용자 정보 삭제 test
2192        service.deleteUser("dooly");
2193        for(UserVO user : service.getUserList()){
2194          System.out.println(user);
2195        }
2196      }
2197
2198    5)결과
2199      UserVO(userId=jimin, name=한지민, gender=여, city=서울)
2200      삭제된 Record with ID = dooly
2201      UserVO(userId=jimin, name=한지민, gender=여, city=서울)
2202
2203
2204
2205  ------------------------------------------------------
2206  Task6. Example of Spring JdbcTemplate
2207  1. Create Table
2208
2209    CREATE TABLE Employee(
2210      id NUMBER(10),
2211      name VARCHAR2(100),
2212      salary NUMBER(10)
2213    );
2214
2215
2216  2. In Package Explorer > right-click > New > Java Project
2217    1)Project Name : JdbcTemplateDemo
2218    2)JRE
2219      -Select [Use default JRE 'jdk-13.0.2' and workspace compiler preferences]
2220    3)Next
2221    4)Uncheck [Create module-info.java file]
2222    5)Finish
2223
2224
2225  3. src > right-click > New > Package
2226    1)Name : com.example
2227    2)Finish
2228
2229
```

4. Java Project를 Spring Project로 변환
  1)JdbcTemplateDemo Project > right-click > Configure > Convert to Maven Project
    -Project : /JdbcTemplateDemo
    -Group Id : JdbcTemplateDemo
    -Artifact Id : JdbcTemplateDemo
    -version : 0.0.1-SNAPSHOT
    -Packaging : jar
    -Finish

  2)JdbcTemplateDemo Project > right-click > Spring > Add Spring Project Nature

  3)pom.xml file에 Spring Context Dependency 추가하기
    <version>0.0.1-SNAPSHOT</version>
    <dependencies>
      <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.2.5.RELEASE</version>
      </dependency>
    </dependencies>

  4)pom.xml > right-click > Run As > Maven install
    [INFO] BUILD SUCCESS 확인

5. Lombok library 추가
  1)https://mvnrepository.com/에서 'lombok'으로 검색
  2)'Project Lombok' click
  3)1.18.12 click
  4)depency copy해서 pom.xml에 붙여넣기

    <dependencies>
      <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.2.5.RELEASE</version>
      </dependency>
      <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
      <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.12</version>
        <scope>provided</scope>
      </dependency>
    </dependencies>

  5)pom.xml > right-click > Run As > Maven install
    [INFO] BUILD SUCCESS 확인

6. pom.xml에 Oracle Jdbc Driver 설정하기
  1)pom.xml에 다음 코드 추가

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

2288
2289 2)pom.xml > right-click > Run As > Maven install
2290 [INFO] BUILD SUCCESS 확인
2291
2292
2293 7. Spring JDBC pom.xml에 추가하기
2294 1)pom.xml에 다음 코드 추가
2295
2296 <dependency>
2297 <groupId>org.springframework</groupId>
2298 <artifactId>spring-jdbc</artifactId>
2299 <version>5.2.5.RELEASE</version>
2300 </dependency>
2301
2302 2)pom.xml > right-click > Run As > Maven install
2303 [INFO] BUILD SUCCESS 확인
2304
2305
2306 8. Employee class 생성
2307 1)com.example > right-click > New > Class
2308 2)Name : Employee
2309 3)Finish
2310
2311 package com.example;
2312
2313 import lombok.AllArgsConstructor;
2314 import lombok.Getter;
2315 import lombok.NoArgsConstructor;
2316 import lombok.Setter;
2317
2318 @Getter
2319 @AllArgsConstructor
2320 @NoArgsConstructor
2321 public class Employee {
2322 @Setter private int id;
2323 private String name;
2324 private float salary;
2325 }
2326
2327
2328 9. EmployeeDao class 생성
2329 1)com.example > right-click > New > Class
2330 2)Name : EmployeeDao
2331 3)Finish
2332
2333 package com.example;
2334
2335 import org.springframework.beans.factory.annotation.Autowired;
2336 import org.springframework.jdbc.core.JdbcTemplate;
2337 import org.springframework.stereotype.Repository;
2338
2339 @Repository
2340 public class EmployeeDao {
2341 @Autowired
2342 private JdbcTemplate jdbcTemplate;
2343
2344 public int saveEmployee(Employee e){
2345 String query="INSERT INTO Employee

```
        VALUES('"+e.getId()+"','"+e.getName()+"','"+e.getSalary()+"')";
2346      return jdbcTemplate.update(query);
2347    }
2348    public int updateEmployee(Employee e){
2349      String query="Update Employee SET
          name='"+e.getName()+"',salary='"+e.getSalary()+"' where id='"+e.getId()+"' ";
2350      return jdbcTemplate.update(query);
2351    }
2352    public int deleteEmployee(Employee e){
2353      String query="DELETE FROM Employee where id='"+e.getId()+"' ";
2354      return jdbcTemplate.update(query);
2355    }
2356  }
2357
2358
2359 10. resources folder 생성하기
2360   1)JdbcTemplateDemo project > right-click > New > Source Folder
2361   2)Folder name : resources
2362   3)Finish
2363
2364
2365 11. resources/dbinfo.properties file 생성
2366
2367    db.driverClass=oracle.jdbc.driver.OracleDriver
2368    db.url=jdbc:oracle:thin:@localhost:1521:XE
2369    db.username=hr
2370    db.password=hr
2371
2372
2373 12. Java Annotation 환경설정 파일 생성
2374   1)com.example > right-click > New > Class
2375   2)Name : ApplicationConfig
2376   3)Finish
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 ApplicationConfig {
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
2426
2427  13. MainClasst Class 생성
2428    1)com.example > right-click > New > Class
2429    2)Name : MainClass
2430    3)Finish
2431
2432      package com.example;
2433
2434      import org.springframework.context.ApplicationContext;
2435      import org.springframework.context.annotation.AnnotationConfigApplicationContext;
2436
2437      public class MainClass {
2438        public static void main(String[] args) {
2439          ApplicationContext ctx=new
              AnnotationConfigApplicationContext(ApplicationConfig.class);
2440
2441          EmployeeDao dao = (EmployeeDao)ctx.getBean("empDao");
2442          int status = dao.saveEmployee(new Employee(102,"Amit",35000));
2443          System.out.println("Insert Status = " + status);
2444
2445          status = dao.updateEmployee(new Employee(102,"Sonoo",15000));
2446          System.out.println("Update Status = " + status);
2447
2448          Employee e=new Employee();
2449          e.setId(102);
2450          status = dao.deleteEmployee(e);
2451          System.out.println("Delete Status = " + status);
2452        }
2453      }
2454
2455    4)결과
2456      Insert Status = 1
2457      Update Status = 1
```

```
2458      Delete Status = 1
2459
2460
2461
2462  ---------------------------------------------------
2463  Task7. Example of PreparedStatement in Spring JdbcTemplate
2464  1. EmployeeDao1 class 생성
2465    1)com.example.EmployeeDao를 복사하여 붙여넣기
2466    2)이름변경 : EmployeeDao1
2467    3)OK
2468
2469    package com.example;
2470    import java.sql.PreparedStatement;
2471    import java.sql.SQLException;
2472
2473    import org.springframework.dao.DataAccessException;
2474    import org.springframework.jdbc.core.JdbcTemplate;
2475    import org.springframework.jdbc.core.PreparedStatementCallback;
2476
2477    public class EmployeeDao {
2478      private JdbcTemplate jdbcTemplate;
2479
2480      public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
2481        this.jdbcTemplate = jdbcTemplate;
2482      }
2483
2484      public Boolean saveEmployeeByPreparedStatement(final Employee e){
2485        String query="INSERT INTO Employee VALUES(?,?,?)";
2486        return jdbcTemplate.execute(query,new PreparedStatementCallback<Boolean>(){
2487          @Override
2488          public Boolean doInPreparedStatement(PreparedStatement ps)
2489              throws SQLException, DataAccessException {
2490
2491            ps.setInt(1,e.getId());
2492            ps.setString(2,e.getName());
2493            ps.setFloat(3,e.getSalary());
2494
2495            return ps.execute();
2496
2497          }
2498        });
2499      }
2500    }
2501
2502  2. ApplicationConfig1 class 생성
2503    1)com.example.ApplicationConfig.java를 복사하여 붙여넣기
2504    2)이름변경 : ApplicationConfig1
2505    3)OK
2506
2507      package com.example;
2508
2509      import javax.sql.DataSource;
2510
2511      import org.springframework.beans.factory.annotation.Value;
2512      import org.springframework.context.annotation.Bean;
2513      import org.springframework.context.annotation.ComponentScan;
2514      import org.springframework.context.support.PropertySourcesPlaceholderConfigurer;
2515      import org.springframework.core.io.ClassPathResource;
```

```
2516      import org.springframework.jdbc.core.JdbcTemplate;
2517      import org.springframework.jdbc.datasource.DriverManagerDataSource;
2518
2519      @ComponentScan(basePackages = "com.example")
2520      public class ApplicationConfig1 {
2521        @Value("${db.driverClass}")
2522        private String driverClassName;
2523        @Value("${db.url}")
2524        private String url;
2525        @Value("${db.username}")
2526        private String username;
2527        @Value("${db.password}")
2528        private String password;
2529
2530        @Bean
2531        public static PropertySourcesPlaceholderConfigurer properties() {
2532          PropertySourcesPlaceholderConfigurer configurer = new
           PropertySourcesPlaceholderConfigurer();
2533          configurer.setLocation(new ClassPathResource("dbinfo.properties"));
2534          return configurer;
2535        }
2536
2537        @Bean
2538        public DataSource dataSource() {
2539          DriverManagerDataSource ds = new DriverManagerDataSource();
2540          ds.setDriverClassName(this.driverClassName);
2541          ds.setUrl(this.url);
2542          ds.setUsername(this.username);
2543          ds.setPassword(this.password);
2544          return ds;
2545        }
2546
2547        @Bean
2548        public JdbcTemplate jdbcTemplate() {
2549          JdbcTemplate template = new JdbcTemplate();
2550          template.setDataSource(this.dataSource());
2551          return template;
2552        }
2553
2554        @Bean
2555        public EmployeeDao1 empDao1() {
2556          EmployeeDao1 empDao1 = new EmployeeDao1();
2557          return empDao1;
2558        }
2559      }
2560
2561
2562  3. MainClass1 class 생성
2563    1)com.example.MainClass 복사하여 붙여넣기
2564    2)이름변경 : MainClass1
2565    3)OK
2566
2567    package com.example;
2568
2569    import org.springframework.context.ApplicationContext;
2570    import org.springframework.context.support.ClassPathXmlApplicationContext;
2571    public class Test {
2572
```

```
2573      public static void main(String[] args) {
2574        ApplicationContext ctx=new ClassPathXmlApplicationContext("applicationContext.xml");

2575
2576        EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");
2577        dao.saveEmployeeByPreparedStatement(new Employee(108,"Amit",35000));
2578      }
2579    }
2580
2581
2582  ----------------------------------------------------
2583  Task8. ResultSetExtractor Example | Fetching Records by Spring JdbcTemplate
2584  1. Employee.java 수정
2585
2586    package com.example;
2587
2588    import lombok.AllArgsConstructor;
2589    import lombok.Getter;
2590    import lombok.NoArgsConstructor;
2591    import lombok.Setter;
2592    import lombok.ToString;
2593
2594    @Getter
2595    @Setter
2596    @AllArgsConstructor
2597    @NoArgsConstructor
2598    @ToString
2599    public class Employee {
2600      private int id;
2601      private String name;
2602      private float salary;
2603    }
2604
2605
2606  2. EmployeeDao2 class 생성
2607    1)com.example.EmployeeDao1를 복사하여 붙여넣기
2608    2)이름변경 : EmployeeDao2
2609    3)OK
2610
2611      package com.example;
2612
2613      import java.sql.ResultSet;
2614      import java.sql.SQLException;
2615      import java.util.ArrayList;
2616      import java.util.List;
2617
2618      import org.springframework.beans.factory.annotation.Autowired;
2619      import org.springframework.dao.DataAccessException;
2620      import org.springframework.jdbc.core.JdbcTemplate;
2621      import org.springframework.jdbc.core.ResultSetExtractor;
2622      import org.springframework.stereotype.Repository;
2623
2624      @Repository
2625      public class EmployeeDao2 {
2626        @Autowired
2627        private JdbcTemplate jdbcTemplate;
2628
2629        public List<Employee> getAllEmployees(){
```

```
2630            return jdbcTemplate.query("SELECT * FROM Employee", new
                ResultSetExtractor<List<Employee>>(){
2631              @Override
2632               public List<Employee> extractData(ResultSet rs) throws SQLException,
2633                   DataAccessException {
2634
2635                 List<Employee> list=new ArrayList<Employee>();
2636                 while(rs.next()){
2637                   Employee e=new Employee();
2638                   e.setId(rs.getInt(1));
2639                   e.setName(rs.getString(2));
2640                   e.setSalary(rs.getInt(3));
2641                   list.add(e);
2642                 }
2643                 return list;
2644               }
2645               });
2646         }
2647       }
2648
2649
2650  3. ApplicationConfig2 class 생성
2651    1)com.example.ApplicationConfig1.java를 복사하여 붙여넣기
2652    2)이름변경 : ApplicationConfig2
2653    3)OK
2654
2655         ...
2656         @Bean
2657         public EmployeeDao2 empDao2() {
2658           EmployeeDao2 empDao2 = new EmployeeDao2();
2659           return empDao2;
2660         }
2661       }
2662
2663
2664  4. MainClass2 class 생성
2665    1)MainClass1.java copy하여 붙여기
2666    2)이름변경 : MainClass2
2667    3)OK
2668
2669       package com.example;
2670
2671       import org.springframework.context.ApplicationContext;
2672       import org.springframework.context.annotation.AnnotationConfigApplicationContext;
2673
2674       public class MainClass2 {
2675         public static void main(String[] args) {
2676           ApplicationContext ctx=new
                AnnotationConfigApplicationContext(ApplicationConfig2.class);
2677
2678           EmployeeDao2 dao2 = (EmployeeDao2)ctx.getBean("empDao2");
2679           dao2.getAllEmployees().forEach(emp -> System.out.println(emp));
2680         }
2681       }
2682
2683    4)결과
2684       Employee(id=108, name=Amit, salary=35000.0)
2685
```

Task9. RowMapper Example | Fetching records by Spring JdbcTemplate

1. EmployeeDao3 class 생성
   1)com.example.EmployeeDao2를 복사하여 붙여넣기
   2)이름변경 : EmployeeDao3
   3)OK

```
package com.example;

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

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.RowMapper;
import org.springframework.stereotype.Repository;

@Repository
public class EmployeeDao3 {
  @Autowired
  private JdbcTemplate jdbcTemplate;

  public List<Employee> getAllEmployeesRowMapper(){
    return jdbcTemplate.query("select * from employee",new RowMapper<Employee>(){

      @Override
      public Employee mapRow(ResultSet rs, int rownumber) throws SQLException {
        Employee e=new Employee();
        e.setId(rs.getInt(1));
        e.setName(rs.getString(2));
        e.setSalary(rs.getInt(3));
        return e;
      }
    });
  }
}
```

2. ApplicationConfig3 class 생성
   1)com.example.ApplicationConfig2.java를 복사하여 붙여넣기
   2)이름변경 : ApplicationConfig3
   3)OK

```
...
@Bean
public EmployeeDao3 empDao3() {
  EmployeeDao3 empDao3 = new EmployeeDao3();
  return empDao3;
}
```

3. MainClass3 class 생성
   1)MainClass2.java copy하여 붙여기
   2)이름변경 : MainClass3
   3)OK

```
2743
2744      package com.example;
2745
2746      import org.springframework.context.ApplicationContext;
2747      import org.springframework.context.annotation.AnnotationConfigApplicationContext;
2748
2749      public class MainClass3 {
2750        public static void main(String[] args) {
2751          ApplicationContext ctx=new
            AnnotationConfigApplicationContext(ApplicationConfig3.class);
2752
2753          EmployeeDao3 dao3 = (EmployeeDao3)ctx.getBean("empDao3");
2754          dao3.getAllEmployeesRowMapper().forEach(emp -> System.out.println(emp));
2755        }
2756      }
2757
2758    4)결과
2759      Employee(id=108, name=Amit, salary=35000.0)
2760
2761
2762
2763  ------------------------------------------------------
2764  Task10. Spring NamedParameterJdbcTemplate Example
2765  1. Create Table
2766
2767    CREATE TABLE Employee(
2768      id NUMBER(10),
2769      name VARCHAR2(100),
2770      salary NUMBER(10)
2771    );
2772
2773
2774  2. In Package Explorer > right-click > New > Java Project
2775    1)Project Name : NamedJdbcTemplateDemo
2776    2)JRE
2777      -Select [Use default JRE 'jdk-13.0.2' and workspace compiler preferences]
2778    3)Next
2779    4)Uncheck [Create module-info.java file]
2780    5)Finish
2781
2782
2783  3. src > right-click > New > Package
2784    1)Name : com.example
2785    2)Finish
2786
2787
2788  4. Java Project를 Spring Project로 변환
2789    1)NamedJdbcTemplateDemo Project > right-click > Configure > Convert to Maven Project
2790      -Project : /NamedJdbcTemplateDemo
2791      -Group Id : NamedJdbcTemplateDemo
2792      -Artifact Id : NamedJdbcTemplateDemo
2793      -version : 0.0.1-SNAPSHOT
2794      -Packaging : jar
2795      -Finish
2796
2797    2)NamedJdbcTemplateDemo Project > right-click > Spring > Add Spring Project Nature
2798
2799    3)pom.xml file에 Spring Context Dependency 추가하기
```

```
2800      <version>0.0.1-SNAPSHOT</version>
2801      <dependencies>
2802        <dependency>
2803          <groupId>org.springframework</groupId>
2804          <artifactId>spring-context</artifactId>
2805          <version>5.2.5.RELEASE</version>
2806        </dependency>
2807      </dependencies>
2808
2809    4)pom.xml > right-click > Run As > Maven install
2810      [INFO] BUILD SUCCESS 확인
2811
2812
2813  5. Lombok library 추가
2814    1)https://mvnrepository.com/에서 'lombok'으로 검색
2815    2)'Project Lombok' click
2816    3)1.18.12 click
2817    4)depency copy해서 pom.xml에 붙여넣기
2818
2819      <dependency>
2820        <groupId>org.projectlombok</groupId>
2821        <artifactId>lombok</artifactId>
2822        <version>1.18.12</version>
2823        <scope>provided</scope>
2824      </dependency>
2825
2826    5)pom.xml > right-click > Run As > Maven install
2827      [INFO] BUILD SUCCESS 확인
2828
2829
2830  6. pom.xml에 Oracle Jdbc Driver 설정하기
2831    1)pom.xml에 다음 코드 추가
2832
2833      <dependency>
2834        <groupId>com.oracle</groupId>
2835        <artifactId>ojdbc8</artifactId>
2836        <version>12.2</version>
2837      </dependency>
2838
2839    2)pom.xml > right-click > Run As > Maven install
2840      [INFO] BUILD SUCCESS 확인
2841
2842
2843  7. Spring JDBC pom.xml에 추가하기
2844    1)pom.xml에 다음 코드 추가
2845
2846      <dependency>
2847         <groupId>org.springframework</groupId>
2848         <artifactId>spring-jdbc</artifactId>
2849         <version>5.2.5.RELEASE</version>
2850      </dependency>
2851
2852    2)pom.xml > right-click > Run As > Maven install
2853      [INFO] BUILD SUCCESS 확인
2854
2855
2856  8. Employee class 생성
2857    1)com.example > right-click > New > Class
```

2)Name : Employee
3)Finish

```
package com.example;

import lombok.AllArgsConstructor;
import lombok.Getter;

@Getter
@AllArgsConstructor
public class Employee {
  private int id;
  private String name;
  private float salary;
}
```

9. EmployeeDao class 생성
1)com.example > New > Class
2)Name : EmployeeDao
3)Finish

```
package com.example;

import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.HashMap;
import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.PreparedStatementCallback;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.stereotype.Repository;

@Repository
public class EmployeeDao {
  @Autowired
  private NamedParameterJdbcTemplate template;

  public  void save (Employee emp){
    String query="INSERT INTO Employee VALUES (:id,:name,:salary)";

    Map<String,Object> map = new HashMap<String,Object>();
    map.put("id", emp.getId());
    map.put("name", emp.getName());
    map.put("salary", emp.getSalary());

    template.execute(query, map, new PreparedStatementCallback<Integer>() {
      @Override
      public Integer doInPreparedStatement(PreparedStatement ps)
          throws SQLException, DataAccessException {
        return ps.executeUpdate();
      }
    });
  }
}
```

10. resources folder 생성하기
  1)NamedJdbcTemplateDemo project > right-click > New > Source Folder
  2)Folder name : resources
  3)Finish

11. resources/dbinfo.properties file 생성

```
db.driverClass=oracle.jdbc.driver.OracleDriver
db.url=jdbc:oracle:thin:@localhost:1521:XE
db.username=hr
db.password=hr
```

12. ApplicationConfig class 생성
  1)com.example > New > Class
  2)Name : ApplicationConfig
  3)Finish

```
package com.example;

import javax.sql.DataSource;

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.support.PropertySourcesPlaceholderConfigurer;
import org.springframework.core.io.ClassPathResource;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.jdbc.datasource.DriverManagerDataSource;

@Configuration
@ComponentScan(basePackages = "com.example")
public class ApplicationConfig {
  @Value("${db.driverClass}")
  private String driverClassName;
  @Value("${db.url}")
  private String url;
  @Value("${db.username}")
  private String username;
  @Value("${db.password}")
  private String password;

  @Bean
  public static PropertySourcesPlaceholderConfigurer properties() {
    PropertySourcesPlaceholderConfigurer configurer = new
    PropertySourcesPlaceholderConfigurer();
    configurer.setLocation(new ClassPathResource("dbinfo.properties"));
    return configurer;
  }

  @Bean
  public DataSource dataSource() {
    DriverManagerDataSource ds = new DriverManagerDataSource();
    ds.setDriverClassName(this.driverClassName);
    ds.setUrl(this.url);
```

```
2973          ds.setUsername(this.username);
2974          ds.setPassword(this.password);
2975          return ds;
2976        }
2977
2978        @Bean
2979        public NamedParameterJdbcTemplate jdbcTemplate() {
2980          NamedParameterJdbcTemplate template = new
              NamedParameterJdbcTemplate(this.dataSource());
2981          return template;
2982        }
2983
2984        @Bean
2985        public EmployeeDao empDao() {
2986          EmployeeDao empDao = new EmployeeDao();
2987          return empDao;
2988        }
2989      }
2990
2991
2992  13. MainClass class 생성
2993    1)com.example > New > Class
2994    2)Name : MainClass
2995    3)Finish
2996
2997    package com.example;
2998
2999    import org.springframework.beans.factory.BeanFactory;
3000    import org.springframework.beans.factory.xml.XmlBeanFactory;
3001    import org.springframework.core.io.ClassPathResource;
3002    import org.springframework.core.io.Resource;
3003
3004    public class SimpleTest {
3005      public static void main(String[] args) {
3006
3007        Resource r=new ClassPathResource("applicationContext.xml");
3008        BeanFactory factory=new XmlBeanFactory(r);
3009
3010        EmpDao dao=(EmpDao)factory.getBean("edao");
3011        dao.save(new Emp(23,"sonoo",50000));
3012
3013      }
3014    }
```