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

```
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
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();
```

```
            if(rs != null) rs.close();
          }catch(SQLException ex) {
            System.out.println(ex);
          }
        }
        return userList;
      }


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

        @Override
        public void insertUser(UserVO user) {
          userDao.insert(user);
        }

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


  3)com.example.test.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)right-click > Run As > Junit Test
  5)결과 -> Junit View에 초록색 bar
      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]


19. JDBC를 이용한 Membership Project - 사용자 정보 수정 test
  1)com.example.dao.UserDaoImplJDBC.java code 수정

      @Override
      public void update(UserVO user) {
        Connection conn = null;
        PreparedStatement pstmt = null;
        try {
          conn = this.dataSource.getConnection();
          String sql = "UPDATE users  SET name = ?, gender = ?, city = ? WHERE userid = ?";
          pstmt = conn.prepareStatement(sql);
          pstmt.setString(1, user.getName());
          pstmt.setString(2, user.getGender());
          pstmt.setString(3, user.getCity());
          pstmt.setString(4, user.getUserId());
          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]



----------------------------------------------------------
Task3. JdbcTemplate를 이용한 Membership Project
1. resources/beans.xml 수정

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

     <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>

     <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
       <property name="dataSource" ref="dataSource" />
     </bean>
   </beans>


2. 사용자 조회 test
   1)com.example.dao.UserDaoImplJDBC.java 복사 후 붙여넣기
   2)이름을 UserDaoImplJdbcTemplate로 변경

     package com.example.dao;

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

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

     import com.example.vo.UserVO;

     @Repository("userDao1")  <---변경
     public class UserDaoImplJdbcTemplate implements UserDao {
       @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      }
1067
1068
1069    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      }
1112
1113
1114    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      }
1123
1124
1125    5)결과
1126      UserVO(userId=jimin, name=한지민, gender=여, city=서울)
1127
1128
1129  3. 사용자 등록 및 목록 조회 test
1130    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(),
          user.getCity());
1136        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
1147  2)com.example.service.UserServiceImpl.java code 수정
1148
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
1160  3)/src/test/java/MembershipTest.java
1161
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
1172  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
1179 4. 사용자 정보 수정 test
1180  1)com.example.dao.UserDaoImplJdbcTemplate.java code 수정
1181
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(),
          user.getCity(),user.getUserId());
1186      System.out.println("갱신된 Record with ID = " + user.getUserId() );
1187    }
1188
1189
1190  2)com.example.service.UserServiceImpl.java code 수정
1191
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        }
```

```
    }


  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
public class UserDaoImpliBatis implements UserDao {

  @Override
  public void insert(UserVO user) {

  }

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

  @Override
  public void update(UserVO user) {

```

```
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        }
1389
1390
1391    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)UserDaoImpliBatis.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 {
```

```
            rd = Resources.getResourceAsReader("SqlMapConfig.xml");
            smc = SqlMapClientBuilder.buildSqlMapClient(rd);
            userList = (List<UserVO>)smc.queryForList("Users.getAll", null);
          } catch (IOException | SQLException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
          }
          return userList;
        }

    3)UserServiceImpl.java

        @Override
        public void insertUser(UserVO user) {
          this.userDao2.insert(user);
        }

        @Override
        public List<UserVO> getUserList() {
          return this.userDao2.readAll();
        }

    4)MembershipTest.java
        @Test
        public void test1() {
          this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
          for(UserVO user : this.service.getUserList()){
            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)Users.xml
        <update id="update" parameterClass="userVO">
          UPDATE USERS
          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

```
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
```

```
6)MyBatis Spring click
7)2.0.4 click
8)dependency를 복사해서 pom.xml에 붙여넣기
  <!-- https://mvnrepository.com/artifact/org.mybatis/mybatis-spring -->
  <dependency>
     <groupId>org.mybatis</groupId>
     <artifactId>mybatis-spring</artifactId>
     <version>2.0.4</version>
  </dependency>

9)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

10)resources/dbinfo.properties
  db.driverClass=oracle.jdbc.driver.OracleDriver
  db.url=jdbc:oracle:thin:@localhost:1521:XE
  db.username=hr
  db.password=hr

11)mybatis-config.xml 생성
  -src > right-click > New > Other > XML > XML File > Next
  -File name : mybatis-config.xml
  -Finish

  -https://github.com/mybatis/mybatis-3/releases
  -mybatis-3.5.4.zip downloads > Unzip
  -mybatis-3.5.4.pdf file 참조

    <?xml version="1.0" encoding="UTF-8"?>
    <!DOCTYPE configuration
          PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
            "http://mybatis.org/dtd/mybatis-3-config.dtd">
    <configuration>
      <properties resource="dbinfo.properties" />
      <typeAliases>
        <typeAlias type="com.example.vo.UserVO" alias="userVO" />
      </typeAliases>
      <environments default="development">
        <environment id="development">
          <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
```

```
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"
          resultMap="userVOResult">
1733        SELECT * FROM USERS WHERE userid = #{id}
1734      </select>
1735
1736    2)com.example.dao.UserDaoImplMyBatis.java 생성
1737      -UserDaoImpliBatis.java를 copy하여 paste
1738      -이름변경 : 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        }
1789
1790    3)UserServiceImpl.java 수정
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) {
```

```
      }
    }

4)MembershipTest.java

    package com.example.test;

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

    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;
    import com.example.vo.UserVO;

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

      @Test
      public void test() {
        UserVO user = this.service.getUser("jimin");
        System.out.println(user);
        assertEquals("한지민", user.getName());
      }
    }

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


3. 사용자 등록 및 목록 조회 test
  1)mybatis-mapper.xml

    <insert id="insert" parameterType="userVO">
      INSERT INTO USERS(userid, name, gender, city)
      VALUES (#{userId}, #{name}, #{gender}, #{city})
    </insert>

    <select id="selectAll" resultType="userVO" resultMap="userVOResult">
      SELECT * FROM USERS
    </select>

  2)UserDaoImplMyBatis.java

    @Override
    public void insert(UserVO user) {
      Reader rd = null;
      SqlSession session = null;
      UserVO userVO = null;
      try {
        rd = Resources.getResourceAsReader("mybatis-config.xml");
        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}
```

```
      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()){
        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)mybatis-mapper.xml

```
<delete id="delete" parameterType="String">
  DELETE FROM USERS WHERE  userid = #{id}
</delete>
```

2)UserDaoImplMyBatis.java

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

3)UserServiceImpl.java

```
@Override
public void deleteUser(String id) {
  this.userDao3.delete(id);
}
```

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()){
    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)



----------------------------------------------------
Task6. Example of Spring JdbcTemplate
1. Create Table

  CREATE TABLE Employee(
    id NUMBER(10),
    name VARCHAR2(100),
    salary NUMBER(10)
  );


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


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


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>
```

```
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)depency 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
```

[INFO] BUILD SUCCESS 확인

8. Employee class 생성
1)com.example > right-click > New > Class
2)Name : Employee
3)Finish

```
package com.example;

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

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

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

```
package com.example;

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

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

  public int saveEmployee(Employee e){
    String query="INSERT INTO Employee
    VALUES('"+e.getId()+"','"+e.getName()+"','"+e.getSalary()+"')";
    return jdbcTemplate.update(query);
  }
  public int updateEmployee(Employee e){
    String query="Update Employee SET
    name='"+e.getName()+"',salary='"+e.getSalary()+"' where id='"+e.getId()+"' ";
    return jdbcTemplate.update(query);
  }
  public int deleteEmployee(Employee e){
    String query="DELETE FROM Employee where id='"+e.getId()+"' ";
    return jdbcTemplate.update(query);
  }
}
```

10. resources folder 생성하기
  1)JdbcTemplateDemo 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. Java Annotation 환경설정 파일 생성
  1)com.example > right-click > 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.support.PropertySourcesPlaceholderConfigurer;
import org.springframework.core.io.ClassPathResource;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.datasource.DriverManagerDataSource;

@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);
    ds.setUsername(this.username);
    ds.setPassword(this.password);
    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. MainClasst 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
          AnnotationConfigApplicationContext(ApplicationConfig.class);
2311
2312          EmployeeDao dao = (EmployeeDao)ctx.getBean("empDao");
2313          int status = dao.saveEmployee(new Employee(102,"Amit",35000));
2314          System.out.println("Insert Status = " + status);
2315
2316          status = dao.updateEmployee(new Employee(102,"Sonoo",15000));
2317          System.out.println("Update Status = " + status);
2318
2319          Employee e=new Employee();
2320          e.setId(102);
2321          status = dao.deleteEmployee(e);
2322          System.out.println("Delete Status = " + status);
2323        }
2324      }
2325
2326    4)결과
2327      Insert Status = 1
2328      Update Status = 1
2329      Delete Status = 1
2330
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
```

```
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.PreparedStatementCallback;

public class EmployeeDao {
  private JdbcTemplate jdbcTemplate;

  public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
    this.jdbcTemplate = jdbcTemplate;
  }

  public Boolean saveEmployeeByPreparedStatement(final Employee e){
    String query="INSERT INTO Employee VALUES(?,?,?)";
    return jdbcTemplate.execute(query,new PreparedStatementCallback<Boolean>(){
      @Override
      public Boolean doInPreparedStatement(PreparedStatement ps)
          throws SQLException, DataAccessException {

        ps.setInt(1,e.getId());
        ps.setString(2,e.getName());
        ps.setFloat(3,e.getSalary());

        return ps.execute();

      }
    });
  }
}
```

2. ApplicationConfig1 class 생성
  1)com.example.ApplicationConfig.java를 복사하여 붙여넣기
  2)이름변경 : ApplicationConfig1
  3)OK

```
  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.support.PropertySourcesPlaceholderConfigurer;
  import org.springframework.core.io.ClassPathResource;
  import org.springframework.jdbc.core.JdbcTemplate;
  import org.springframework.jdbc.datasource.DriverManagerDataSource;

  @ComponentScan(basePackages = "com.example")
  public class ApplicationConfig1 {
    @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
```

```
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
                ResultSetExtractor<List<Employee>>(){
2502            @Override
2503             public List<Employee> extractData(ResultSet rs) throws SQLException,
2504                 DataAccessException {
2505
2506              List<Employee> list=new ArrayList<Employee>();
2507              while(rs.next()){
2508                Employee e=new Employee();
2509                e.setId(rs.getInt(1));
2510                e.setName(rs.getString(2));
2511                e.setSalary(rs.getInt(3));
2512                list.add(e);
2513              }
2514              return list;
```

```
            }
            });
        }
    }


3. ApplicationConfig2 class 생성
  1)com.example.ApplicationConfig1.java를 복사하여 붙여넣기
  2)이름변경 : ApplicationConfig2
  3)OK

      ...
      @Bean
      public EmployeeDao2 empDao2() {
        EmployeeDao2 empDao2 = new EmployeeDao2();
        return empDao2;
      }
    }


4. MainClass2 class 생성
  1)MainClass1.java copy하여 붙여기
  2)이름변경 : MainClass2
  3)OK

    package com.example;

    import org.springframework.context.ApplicationContext;
    import org.springframework.context.annotation.AnnotationConfigApplicationContext;

    public class MainClass2 {
      public static void main(String[] args) {
        ApplicationContext ctx=new
        AnnotationConfigApplicationContext(ApplicationConfig2.class);

        EmployeeDao2 dao2 = (EmployeeDao2)ctx.getBean("empDao2");
        dao2.getAllEmployees().forEach(emp -> System.out.println(emp));
      }
    }

  4)결과
    Employee(id=108, name=Amit, salary=35000.0)



-----------------------------------------------------
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;

```

```
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            @Override
2585            public Employee mapRow(ResultSet rs, int rownumber) throws SQLException {
2586              Employee e=new Employee();
2587              e.setId(rs.getInt(1));
2588              e.setName(rs.getString(2));
2589              e.setSalary(rs.getInt(3));
2590              return e;
2591            }
2592          });
2593        }
2594      }
2595
2596
2597  2. ApplicationConfig3 class 생성
2598    1)com.example.ApplicationConfig2.java를 복사하여 붙여넣기
2599    2)이름변경 : ApplicationConfig3
2600    3)OK
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 붙여기
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
            AnnotationConfigApplicationContext(ApplicationConfig3.class);
2623
2624          EmployeeDao3 dao3 = (EmployeeDao3)ctx.getBean("empDao3");
2625          dao3.getAllEmployeesRowMapper().forEach(emp -> System.out.println(emp));
2626        }
2627      }
```

```
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)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

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>
```

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

7. Spring JDBC pom.xml에 추가하기
1)pom.xml에 다음 코드 추가

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

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

8. Employee class 생성
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);
    ds.setUsername(this.username);
    ds.setPassword(this.password);
    return ds;
  }

  @Bean
  public NamedParameterJdbcTemplate jdbcTemplate() {
    NamedParameterJdbcTemplate template = new
    NamedParameterJdbcTemplate(this.dataSource());
    return template;
  }

  @Bean
  public EmployeeDao empDao() {
    EmployeeDao empDao = new EmployeeDao();
```

```
          return empDao;
        }
      }


13. MainClass class 생성
  1)com.example > New > Class
  2)Name : MainClass
  3)Finish

  package com.example;

  import org.springframework.beans.factory.BeanFactory;
  import org.springframework.beans.factory.xml.XmlBeanFactory;
  import org.springframework.core.io.ClassPathResource;
  import org.springframework.core.io.Resource;

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

      Resource r=new ClassPathResource("applicationContext.xml");
      BeanFactory factory=new XmlBeanFactory(r);

      EmpDao dao=(EmpDao)factory.getBean("edao");
      dao.save(new Emp(23,"sonoo",50000));

    }
  }
```