HOL : Spring JDBC

Task1. Spring Jdbc Demo
1. SpringJdbcDemo project 생성
  1)New > Java Project >
  2)Project name : SpringJdbcDemo > Finish

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

3. config folder 생성
  1)SpringJdbcDemo project > right-click > Build Path > Coinfigure Build Path
  2)Source Tab > Add Folder > Select SpringJdbcDemo project > Click [Create New Folder] button
  3)Folder name : config > Finish > OK

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

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

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

```
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

```
<!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
<dependency>
    <groupId>org.springframework</groupId>
    <artifactId>spring-context</artifactId>
    <version>5.2.0.RELEASE</version>
</dependency>
```

9. Spring JDBC 설치
1)JdbcTemplate를 사용하기 위해 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>5.2.0.RELEASE</version>
</dependency>
```

10. 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>
```

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

12. Bean Configuration XML 작성
1)/src/config > right-click > New > Other > Spring > Spring Bean Configuration File
2)File name : beans.xml > Next
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.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>

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
   oracle.jdbc.driver.T4CConnection@51c8530f


-------------------------------------------------
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

3. /src > right-click > New > Package
   1)Package name : com.example.vo

```

2)com.example.vo.UserVO.java 생성

```
package com.example.vo;

public class UserVO {

    private String userId;
    private String name;
    private String gender;
    private String city;

    public UserVO() {}

    public UserVO(String userId, String name, String gender, String city) {
        this.userId = userId;
        this.name = name;
        this.gender = gender;
        this.city = city;
    }

    public String getUserId() {
        return userId;
    }

    public void setUserId(String userId) {
        this.userId = userId;
    }

    public String getName() {
        return name;
    }

    public void setName(String name) {
        this.name = name;
    }

    public String getGender() {
        return gender;
    }

    public void setGender(String gender) {
        this.gender = gender;
    }

    public String getCity() {
        return city;
    }

    public void setCity(String city) {
        this.city = city;
    }

```

```
208         @Override
209         public String toString() {
210            return "User [userId=" + userId + ", name=" + name + ", gender="
211                  + gender + ", city=" + city + "]";
212         }
213      }
214
215 4. /src > right-click > New > Package
216    1)Package name : com.example.service
217
218    2)UserService interface 생성
219       -com.example.service.UserService.java
220
221       package com.example.service;
222
223       import java.util.List;
224       import com.example.vo.UserVO;
225
226       public interface UserService {
227
228          void insertUser(UserVO user);
229
230          List<UserVO> getUserList();
231
232          void deleteUser(String id);
233
234          UserVO getUser(String id);
235
236          void updateUser(UserVO user);
237       }
238
239    3)UserServiceImpl class 생성
240       -com.example.service.UserServiceImpl.java
241          package com.example.service;
242
243          import java.util.List;
244          import com.example.vo.UserVO;
245
246          public class UserServiceImpl implements UserService {
247
248             @Override
249             public void insertUser(UserVO user) {
250                // TODO Auto-generated method stub
251
252             }
253
254             @Override
255             public List<UserVO> getUserList() {
256                // TODO Auto-generated method stub
257                return null;
258             }
259
```

```
260             @Override
261             public void deleteUser(String id) {
262                 // TODO Auto-generated method stub
263
264             }
265
266             @Override
267             public UserVO getUser(String id) {
268                 // TODO Auto-generated method stub
269                 return null;
270             }
271
272             @Override
273             public void updateUser(UserVO user) {
274                 // TODO Auto-generated method stub
275
276             }
277         }
278
279 5. /src > right-click > New > Package
280    1)Package name : com.example.dao
281
282    2)UserDao interface 생성
283       -com.example.dao.UserDao.java
284
285          package com.example.dao;
286
287          import java.util.List;
288          import com.example.vo.UserVO;
289
290          public interface UserDao {
291             void insert(UserVO user);
292
293             List<UserVO> readAll();
294
295             void update(UserVO user);
296
297             void delete(String id);
298
299             UserVO read(String id);
300          }
301
302    3)UserDaoImplJDBC class 생성
303       -com.example.dao.UserDaoImplJDBC.java
304          package com.example.dao;
305
306          import java.util.List;
307          import com.example.vo.UserVO;
308
309          public class UserDaoImplJDBC implements UserDao {
310
311             @Override
```

```
312             public void insert(UserVO user) {
313                // TODO Auto-generated method stub
314
315             }
316
317             @Override
318             public List<UserVO> readAll() {
319                // TODO Auto-generated method stub
320                return null;
321             }
322
323             @Override
324             public void update(UserVO user) {
325                // TODO Auto-generated method stub
326
327             }
328
329             @Override
330             public void delete(String id) {
331                // TODO Auto-generated method stub
332
333             }
334
335             @Override
336             public UserVO read(String id) {
337                // TODO Auto-generated method stub
338                return null;
339             }
340          }
341
342  6. Java Project를 Spring Project로 변환
343     1)Membership Project > right-click > Configure > Convert to Maven Project
344        -Project : /Membership
345        -Group Id : Membership
346        -Artifact Id : Membership
347        -version : 0.0.1-SNAPSHOT
348        -Packaging : jar
349        -Finish
350
351     2)Membership Project > right-click > Spring > Add Spring Project Nature
352
353  7. pom.xml 파일에 Spring Context Dependency 추가하기
354     <version>0.0.1-SNAPSHOT</version>
355     <dependencies>
356        <dependency>
357           <groupId>org.springframework</groupId>
358           <artifactId>spring-context</artifactId>
359           <version>5.2.0.RELEASE</version>
360        </dependency>
361     </dependencies>
362
363     -pom.xml > right-click > Run As > Maven install
```

364 [INFO] BUILD SUCCESS 확인
365
366 8. Oracle Jdbc Driver 설치
367 1)Oracle 12C 인 경우
368 <dependency>
369 <groupId>com.oracle</groupId>
370 <artifactId>ojdbc8</artifactId>
371 <version>12.2</version>
372 </dependency>
373
374 2)Oracle 11g 인 경우
375 <dependency>
376 <groupId>com.oracle</groupId>
377 <artifactId>ojdbc6</artifactId>
378 <version>11.2</version>
379 </dependency>
380
381 <참고>
382 -MySQL일 경우에는 'spring mysql'로 검색하여 MySQL Connector/J를 설치한다.
383 <!-- https://mvnrepository.com/artifact/mysql/mysql-connector-java -->
384 <dependency>
385 <groupId>mysql</groupId>
386 <artifactId>mysql-connector-java</artifactId>
387 <version>8.0.18</version>
388 </dependency>
389
390 9. Spring JDBC 설치
391 1)JdbcTemplate를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.
392
393 <dependency>
394 <groupId>org.springframework</groupId>
395 <artifactId>spring-jdbc</artifactId>
396 <version>5.2.0.RELEASE</version>
397 </dependency>
398
399 2)pom.xml에 붙여 넣고 Maven Install 하기
400 [INFO] BUILD SUCCESS 확인
401
402
403 10. resource folder 생성
404 1)Membership project > right-click > Build Path > Coinfigure Build Path
405 2)Source Tab > Add Folder > Select Membership project > Click [Create New Folder] button
406 3)Folder name : resources > Finish > OK
407 4)Apply and Close
408
409 11. dbinfo.properties 파일 생성
410 1)/resources > right-click > New > File
411 2)File name : dbinfo.properties > Finish
412
413 db.driverClass=oracle.jdbc.driver.OracleDriver
414 db.url=jdbc:oracle:thin:@localhost:1521:XE
415 db.username=hr

```
416     db.password=hr
417
418   <참고>
419   3)MySQL일 경우에는 다음과 같이 설정한다.
420     db.driverClass=com.mysql.jdbc.Driver
421     db.url=jdbc:mysql://192.168.136.5:3306/world
422     db.username=root
423     db.password=javamysql
424
425 12. Bean Configuration XML 작성
426   1)/resources > right-click > New > Spring > Spring Bean Configuration File
427   2)File name : beans.xml > Finish
428   3)Namespace Tab
429   4)Check context - http://www.springframework.org/schema/context
430
431     <?xml version="1.0" encoding="UTF-8"?>
432     <beans xmlns="http://www.springframework.org/schema/beans"
433       xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
434       xmlns:context="http://www.springframework.org/schema/context"
435       xsi:schemaLocation="http://www.springframework.org/schema/beans
          http://www.springframework.org/schema/beans/spring-beans.xsd
436         http://www.springframework.org/schema/context
            http://www.springframework.org/schema/context/spring-context-3.2.xsd">
437
438       <context:property-placeholder location="classpath:dbinfo.properties" />
439       <bean id="dataSource" class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
440         <property name="driverClass" value="${db.driverClass}" />
441         <property name="url" value="${db.url}" />
442         <property name="username" value="${db.username}" />
443         <property name="password" value="${db.password}" />
444       </bean>
445     </beans>
446
447 13. Membership Project의 Bean 등록 및 의존 관계 설정
448   1)<context:component-scan> tag 사용
449   2)@Service, @Repository annotation을 선언한 class들과 @Autowired annotation을 선언하여 의존관계를
      설정한 class들이 위치한 package를 Scan하기 위한 설정을 XML에 해주어야 한다.
450   3)beans.xml에 다음 code를 추가한다.
451
452     <context:component-scan base-package="com.example" />
453
454 14. Spring TestContext Framework 사용하기
455   1)MVN Repository에서 'spring test'로 검색하여 'Spring TextContext Framework'을 pom.xml에 추가
456     <!-- https://mvnrepository.com/artifact/org.springframework/spring-test -->
457     <dependency>
458       <groupId>org.springframework</groupId>
459       <artifactId>spring-test</artifactId>
460       <version>5.2.0.RELEASE</version>
461       <scope>test</scope>
462     </dependency>
463
464   2)MVN Repository에서 'junit'로 검색하여 'JUnit 4.12'을 pom.xml에 추가
```

```
465     <!-- https://mvnrepository.com/artifact/junit/junit -->
466     <dependency>
467         <groupId>junit</groupId>
468         <artifactId>junit</artifactId>
469         <version>4.12</version>
470         <scope>test</scope>
471     </dependency>
472
473  3)pom.xml Maven Install 하기
474     [INFO] BUILD SUCCESS 확인
475
476  4)/src > right-click > New > Package
477  5)Package Name : com.example.test
478  6)com.example.test > right-click > New > JUnit Test Case
479  7)Select [New JUnit 4 test]
480  8)Name : MembershipTest > Finish
481  9)Select [Not now] > OK
482
483     package com.example.test;
484
485     import org.junit.Test;
486     import org.junit.runner.RunWith;
487     import org.springframework.beans.factory.annotation.Autowired;
488     import org.springframework.test.context.ContextConfiguration;
489     import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
490
491     import com.example.service.UserService;
492
493     @RunWith(SpringJUnit4ClassRunner.class)
494     @ContextConfiguration(locations="classpath:beans.xml")
495     public class MembershipTest {
496
497         @Autowired
498         UserService service;
499
500         @Test
501         public void test() {
502
503         }
504     }
505
506  10)만일 해당 객체를 찾을 수 없다는 오류가 계속 발생하면
507     -해당 Project > right-click > Build Path > Libraries tab
508     -spring-test-5.2.0.RELEASE.jar 선택 후 [Remove] 로 삭제
509     -[Add External JARs...] Click
510     -Local M2 Repository(e.g
        C:₩Users₩bluee₩.m2₩repository₩org₩springframework₩spring-test₩5.2.0.RELEASE)에서 직접 jar를
        선택할 것
511     -[Order and Export] tab에서 spring-test-5.2.0.RELEASE.jar 선택 후 [Up] button을 클릭
512     -해당 Project/src 바로 아래까지 올리고 [Apply and Close] Click
513
514
```

15. JDBC를 이용한 Membership Project
1)사용자 조회 test
-com.example.dao.UserDaoImplJDBC.java 수정

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

   @Autowired
   public void setDataSource(DataSource dataSource) {
      this.dataSource = dataSource;
   }

   ...
   @Override
   public UserVO read(String id) {
      Connection conn = null;
      PreparedStatement pstmt = null;
      ResultSet rs = null;
      UserVO userVO = null;
      try {
         conn = this.dataSource.getConnection();
         pstmt = conn.prepareStatement("SELECT * FROM users WHERE userid = ?");
         pstmt.setString(1, id);
         rs = pstmt.executeQuery();
         rs.next();
         userVO = new UserVO(rs.getString("userid"), rs.getString("name"), rs.getString("gender"),
         rs.getString("city"));
      }catch(SQLException ex) {
         System.out.println(ex);
      }finally {
         try {
            if(conn != null) conn.close();
            if(pstmt != null) pstmt.close();
            if(rs != null) rs.close();
         }catch(SQLException ex) {
            System.out.println(ex);
         }
      }
      return userVO;
   }
```

-com.example.service.UserServiceImpl.java 수정

```
@Service("userService")
public class UserServiceImpl implements UserService {

   @Autowired
   UserDao userDao;

   ...
   @Override
```

```
566             public UserVO getUser(String id) {
567                return userDao.read(id);
568             }
569
570          -com.example.test.MembershipTest.java
571
572             @Test
573             public void test() {
574                //사용자 조회 test
575                UserVO user = service.getUser("jimin");
576                System.out.println(user);
577                assertEquals("한지민", user.getName());
578             }
579
580          -right-click > Run As > Junit Test
581          -결과 -> Junit View에 초록색 bar
582             UserVO [userId=jimin, name=한지민, gender=여, city=서울]
583
584       2)사용자 등록 및 목록 조회 test
585          -com.example.dao.UserDaoImplJDBC.java code 수정
586             @Override
587             public void insert(UserVO user) {
588                Connection conn = null;
589                PreparedStatement pstmt = null;
590                try {
591                   conn = this.dataSource.getConnection();
592                   String sql = "INSERT INTO users (userid, name, gender,city) VALUES (?, ?, ?, ?)";
593                   pstmt = conn.prepareStatement(sql);
594                   pstmt.setString(1, user.getUserId());
595                   pstmt.setString(2, user.getName());
596                   pstmt.setString(3, user.getGender());
597                   pstmt.setString(4, user.getCity());
598                   pstmt.executeUpdate();
599                   System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" + user.getName());
600                }catch(SQLException ex) {
601                   System.out.println(ex);
602                }finally {
603                   try {
604                      if(conn != null) conn.close();
605                      if(pstmt != null) pstmt.close();
606                   }catch(SQLException ex) {
607                      System.out.println(ex);
608                   }
609                }
610             }
611
612             @Override
613             public List<UserVO> readAll() {
614                Connection conn = null;
615                Statement stmt = null;
616                ResultSet rs = null;
617                List<UserVO> userList = null;
```

```
618             try {
619                conn = this.dataSource.getConnection();
620                stmt = conn.createStatement();
621                rs = stmt.executeQuery("SELECT * FROM users");
622                userList = new ArrayList<UserVO>();
623                while(rs.next()) {
624                   UserVO userVO = new UserVO(rs.getString("userid"), rs.getString("name"),
                      rs.getString("gender"), rs.getString("city"));
625                   userList.add(userVO);
626                }
627             }catch(SQLException ex) {
628                System.out.println(ex);
629             }finally {
630                try {
631                   if(conn != null) conn.close();
632                   if(stmt != null) stmt.close();
633                   if(rs != null) rs.close();
634                }catch(SQLException ex) {
635                   System.out.println(ex);
636                }
637             }
638             return userList;
639          }
640
641       -com.example.service.UserServiceImpl.java code 수정
642
643             @Override
644             public void insertUser(UserVO user) {
645                userDao.insert(user);
646             }
647
648             @Override
649             public List<UserVO> getUserList() {
650                return userDao.readAll();
651             }
652
653       -com.example.test.MembershipTest.java
654
655             ...
656             @Test
657             public void test1() {
658                //사용자 등록 및 목록조회 test
659                this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
660                for(UserVO user : this.service.getUserList()){
661                   System.out.println(user);
662                }
663             }
664
665       -right-click > Run As > Junit Test
666       -결과 -> Junit View에 초록색 bar
667       UserVO [userId=jimin, name=한지민, gender=여, city=서울]
668       등록된 Record UserId=dooly Name=둘리
```

```
669     UserVO [userId=dooly, name=둘리, gender=남, city=경기]
670     UserVO [userId=jimin, name=한지민, gender=여, city=서울]
671
672
673  3)사용자 정보 수정 test
674     -com.example.dao.UserDaoImplJDBC.java code 수정
675
676        @Override
677        public void update(UserVO user) {
678           Connection conn = null;
679           PreparedStatement pstmt = null;
680           try {
681              conn = this.dataSource.getConnection();
682              String sql = "UPDATE users  SET name = ?, gender = ?, city = ? WHERE userid = ?";
683              pstmt = conn.prepareStatement(sql);
684              pstmt.setString(1, user.getName());
685              pstmt.setString(2, user.getGender());
686              pstmt.setString(3, user.getCity());
687              pstmt.setString(4, user.getUserId());
688              pstmt.executeUpdate();
689              System.out.println("갱신된 Record with ID = " + user.getUserId() );
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              }catch(SQLException ex) {
697                 System.out.println(ex);
698              }
699           }
700        }
701
702     -com.example.service.UserServiceImpl.java code 수정
703
704           @Override
705           public void updateUser(UserVO user) {
706              userDao.update(user);
707           }
708
709     -com.example.test.MembershipTest.java
710
711           @Ignore @Test
712           public void test1() {
713              //사용자 등록 및 목록조회 test
714              this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
715              for(UserVO user : this.service.getUserList()){
716                 System.out.println(user);
717              }
718           }
719
720           @Test
```

```
721             public void test2() {
722                //사용자 정보 수정 test
723                service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
724                UserVO user = service.getUser("dooly");
725                System.out.println(user);
726             }
727
728       -right-click > Run As > Junit Test
729       -결과 -> Junit View에 초록색 bar
730       UserVO [userId=jimin, name=한지민, gender=여, city=서울]
731       갱신된 Record with ID = dooly
732       UserVO [userId=dooly, name=김둘리, gender=여, city=부산]
733
734    4)사용자 정보 삭제 test
735       -com.example.dao.UserDaoImplJDBC.java code 수정
736
737          @Override
738          public void delete(String id) {
739             Connection conn = null;
740             PreparedStatement pstmt = null;
741             try {
742                conn = this.dataSource.getConnection();
743                pstmt = conn.prepareStatement("DELETE FROM users WHERE userid = ?");
744                pstmt.setString(1, id);
745                pstmt.executeUpdate();
746                System.out.println("삭제된 Record with ID = " + id );
747             }catch(SQLException ex) {
748                System.out.println(ex);
749             }finally {
750                try {
751                   if(conn != null) conn.close();
752                   if(pstmt != null) pstmt.close();
753                }catch(SQLException ex) {
754                   System.out.println(ex);
755                }
756             }
757          }
758
759       -com.example.service.UserServiceImpl.java code 수정
760
761          @Override
762          public void deleteUser(String id) {
763             userDao.delete(id);
764          }
765
766       -com.example.test.MembershipTest.java
767             @Test
768             public void test() {
769                UserVO user = this.service.getUser("jimin");
770                System.out.println(user);
771                assertEquals("한지민", user.getName());
772             }
```

```
773             @Ignore @Test
774             public void test1() {
775                this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
776                for(UserVO user : this.service.getUserList()){
777                   System.out.println(user);
778                }
779             }
780
781             @Ignore @Test
782             public void test2() {
783                service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
784                UserVO user = service.getUser("dooly");
785                System.out.println(user);
786             }
787
788             @Test
789             public void test3() {
790                //사용자 정보 삭제 test
791                service.deleteUser("dooly");
792                for(UserVO user : service.getUserList()){
793                   System.out.println(user);
794                }
795             }
796
797       -right-click > Run As > Junit Test
798       -결과 -> Junit View에 초록색 bar
799       UserVO [userId=jimin, name=한지민, gender=여, city=서울]
800       삭제된 Record with ID = dooly
801       UserVO [userId=jimin, name=한지민, gender=여, city=서울]
802
803
804  ------------------------------------------------------------
805  Task3. JdbcTemplate를 이용한 Membership Project
806  1. 사용자 조회 test
807     1)com.example.dao.UserDaoImplJDBC.java 복사 후 붙여넣기
808     2)이름을 UserDaoImplJdbcTemplate.java로 변경
809
810       @Repository("userDao1")     <---변경
811       public class UserDaoImplJdbcTemplate implements UserDao {
812          private JdbcTemplate jdbcTemplate;     <---변경
813
814          @Autowired
815          public void setDataSource(DataSource dataSource) {
816             this.jdbcTemplate = new JdbcTemplate(dataSource);  <--- 변경
817          }
818
819          class UserMapper implements RowMapper<UserVO> {
820             public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
821                UserVO user = new UserVO();
822                user.setUserId(rs.getString("userid"));
823                user.setName(rs.getString("name"));
824                user.setGender(rs.getString("gender"));
```

```
825                 user.setCity(rs.getString("city"));
826                 return user;
827              }
828           }
829
830           ...
831           @Override
832           public UserVO read(String id) {
833              String SQL = "SELECT * FROM users WHERE userid = ?";
834              try {
835                 UserVO user = jdbcTemplate.queryForObject(SQL,
836                       new Object[] { id }, new UserMapper());
837                 return user;
838              }catch(EmptyResultDataAccessException e){
839                 return null;
840              }
841           }
842
843     3)com.example.service.UserServiceImpl.java 수정
844
845        @Service("userService")
846        public class UserServiceImpl implements UserService {
847
848           @Autowired
849           UserDao userdao1;      <---변경
850
851           ...
852           @Override
853           public UserVO getUser(String id) {
854              return userdao1.read(id);
855           }
856
857     4)/src/test/java/MembershipTest.java
858
859        @Test
860        public void test() {
861           //사용자 조회 test
862           UserVO user = service.getUser("jimin");
863           System.out.println(user);
864           assertEquals("한지민", user.getName());
865        }
866
867  2. 사용자 등록 및 목록 조회 test
868     1)com.example.daoUserDaoImplJdbcTemplate.java code 수정
869
870        @Override
871        public void insert(UserVO user) {
872           String SQL = "INSERT INTO users (userid, name, gender,city) VALUES (?, ?, ?, ?)";
873           jdbcTemplate.update(SQL, user.getUserId(), user.getName(), user.getGender(), user.getCity());
874           System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" + user.getName());
875        }
876
```

```
877         @Override
878         public List<UserVO> readAll() {
879             String SQL = "SELECT * FROM users";
880             List<UserVO>  userList = jdbcTemplate.query(SQL, new UserMapper());
881             return userList;
882         }
883
884     2)com.example.service.UserServiceImpl.java code 수정
885
886         @Override
887         public void insertUser(UserVO user) {
888             userDao3.insert(user);
889         }
890
891         @Override
892         public List<UserVO> getUserList() {
893             return userDao3.readAll();
894         }
895
896     3)/src/test/java/MembershipTest.java
897
898         @Test
899         public void test1() {
900             //사용자 등록 및 목록조회 test
901             this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
902             for(UserVO user : this.service.getUserList()){
903                 System.out.println(user);
904             }
905         }
906
907 3. 사용자 정보 수정 test
908     1)com.example.dao.UserDaoImplJdbcTemplate.java code 수정
909
910         @Override
911         public void update(UserVO user) {
912             String SQL = "UPDATE users  SET name = ?, gender = ?, city = ? WHERE userid = ?";
913             jdbcTemplate.update(SQL, user.getName(), user.getGender(), user.getCity(),user.getUserId());
914             System.out.println("갱신된 Record with ID = " + user.getUserId() );
915         }
916
917     2)com.example.service.UserServiceImpl.java code 수정
918
919         @Override
920         public void updateUser(UserVO user) {
921             userDao3.update(user);
922         }
923
924     3)/src/test/java/MembershipTest.java
925
926         @Ignore @Test
927         public void test1() {
928             //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. 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) {
        userDao3.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);
        }
    }
```

Task4. iBATIS를 이용한 Membership Project

1. 준비

1)mvnrepository(https://mvnrepository.com에서 'ibatis'로 검색

2)ibatis Sqlmap에서 2.3.4.726으로 들어가서 아래의 code를 복사해서 pom.xml에 넣기

```
    <dependency>
        <groupId>org.apache.ibatis</groupId>
        <artifactId>ibatis-sqlmap</artifactId>
        <version>2.3.4.726</version>
```

```
    </dependency>

3)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

4)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>

5)User.xml 파일 생성
  -com.example.dao/Users.xml
    <?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#
```

```
1032        </select>
1033
1034    2)com.example.dao.UserDaoImpliBatis.java 생성
1035        @Repository("userDao2")  <---변경
1036        public class UserDaoImpliBatis implements UserDao {
1037            @Override
1038            public UserVO read(String id) {
1039                Reader rd = null;
1040                SqlMapClient smc = null;
1041                UserVO userVO = null;
1042                try {
1043                    rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1044                    smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1045                    userVO = (UserVO)smc.queryForObject("Users.useResultMap", id);
1046                } catch (IOException | SQLException e) {
1047                    // TODO Auto-generated catch block
1048                    e.printStackTrace();
1049                }
1050                return userVO;
1051            }
1052        }
1053
1054    3)com.example.service.UserServiceImpl.java 수정
1055
1056        @Service("userService")
1057        public class UserServiceImpl implements UserService {
1058
1059            @Autowired
1060            UserDao userDao2;    //userDao2로 변경
1061
1062            ...
1063            @Override
1064            public UserVO getUser(String id) {
1065                return userDao2.read(id);
1066            }
1067
1068    4)/src/test/java/MembershipTest.java
1069
1070        @Test
1071        public void test() {
1072            //사용자 조회 test
1073            UserVO user = service.getUser("jimin");
1074            System.out.println(user);
1075            assertEquals("한지민", user.getName());
1076        }
1077
1078 3. 사용자 등록 및 목록 조회 test
1079    1)Users.xml
1080        <insert id="insert" parameterClass="userVO">
1081            INSERT INTO USERS(userid, name, gender, city)
1082            VALUES (#userId#, #name#, #gender#, #city#)
1083        </insert>
```

```
1084
1085        <select id="getAll" resultClass="userVO">
1086          SELECT * FROM USERS
1087        </select>
1088
1089      2)UserDaoImpliBatis.java
1090        @Override
1091        public void insert(UserVO user) {
1092          Reader rd = null;
1093          SqlMapClient smc = null;
1094          UserVO userVO = null;
1095          try {
1096            rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1097            smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1098            smc.insert("Users.insert", user);
1099            System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" + user.getName());
1100          } catch (IOException | SQLException e) {
1101            // TODO Auto-generated catch block
1102            e.printStackTrace();
1103          }
1104        }
1105
1106        @Override
1107        public List<UserVO> readAll() {
1108          Reader rd = null;
1109          SqlMapClient smc = null;
1110          List<UserVO> userList = null;
1111          try {
1112            rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1113            smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1114            userList = (List<UserVO>)smc.queryForList("Users.getAll", null);
1115          } catch (IOException | SQLException e) {
1116            // TODO Auto-generated catch block
1117            e.printStackTrace();
1118          }
1119          return userList;
1120        }
1121
1122      3)MembershipTest.java
1123        @Test
1124        public void test1() {
1125          this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1126          for(UserVO user : this.service.getUserList()){
1127            System.out.println(user);
1128          }
1129        }
1130
1131    4. 사용자 정보 수정 test
1132      1)Users.xml
1133        <update id="update" parameterClass="userVO">
1134          UPDATE USERS
1135          SET     name = #name#, gender = #gender#, city = #city#
```

```
1136          WHERE  userId = #userId#
1137       </update>
1138
1139    2)com.example.dao.UserDaoImpliBatis.java code 수정
1140       @Override
1141       public void update(UserVO user) {
1142          Reader rd = null;
1143          SqlMapClient smc = null;
1144          UserVO userVO = null;
1145          try {
1146             rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1147             smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1148             smc.update("Users.update", user);
1149             System.out.println("갱신된 Record with ID = " + user.getUserId() );
1150          } catch (IOException | SQLException e) {
1151             // TODO Auto-generated catch block
1152             e.printStackTrace();
1153          }
1154       }
1155
1156    3)MembershipTest.java 수정
1157       @Ignore @Test
1158       public void test1() {
1159          this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1160          for(UserVO user : this.service.getUserList()){
1161             System.out.println(user);
1162          }
1163       }
1164
1165       @Test
1166       public void test2() {
1167          service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1168          UserVO user = service.getUser("dooly");
1169          System.out.println(user);
1170       }
1171
1172 5. 사용자 정보 삭제 test
1173    1)Users.xml
1174       <delete id="delete" parameterClass="String">
1175          DELETE FROM USERS WHERE  userid = #id#
1176       </delete>
1177
1178    2)com.example.dao.UserDaoImpliBatis.java code 수정
1179       @Override
1180       public void delete(String id) {
1181          Reader rd = null;
1182          SqlMapClient smc = null;
1183          UserVO userVO = null;
1184          try {
1185             rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1186             smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1187             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)MembershipTest.java 수정
      @Test
      public void test() {
         UserVO user = this.service.getUser("jimin");
         System.out.println(user);
         assertEquals("한지민", user.getName());
      }
      @Ignore @Test
      public void test1() {
         this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
         for(UserVO user : this.service.getUserList()){
            System.out.println(user);
         }
      }

      @Ignore @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);
         }
      }


---------------------------------------------------------------
Task5. MyBatis를 이용한 Membership Project
1. 준비
   1)mvnrepository(https://mvnrepository.com에서 'Mybatis'로 검색
   2)MyBatis에서 3.5.3로 들어가서 아래의 code를 복사해서 pom.xml에 붙여넣기
      <dependency>
         <groupId>org.mybatis</groupId>
         <artifactId>mybatis</artifactId>
         <version>3.5.3</version>
      </dependency>

   3)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

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

5)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-3.5.3.zip downloads > Unzip
-mybatis-3.5.3.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>
```

6)com.example.dao/mybatis-mapper.xml 생성

```
<?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)UserServiceImpl.java 수정

```
@Service("userService")
public class UserServiceImpl implements UserService {

  @Autowired
  UserDao userDao3;
```

3)com.example.dao.UserDaoImplMyBatis.java 생성

```
  @Repository("userDao3")
  public class UserDaoImplMyBatis implements UserDao {
    ...
    @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) {
        // TODO Auto-generated catch block
        e.printStackTrace();
      }
      return userVO;
    }
```

4)MembershipTest.java

```
@RunWith(SpringJUnit4ClassRunner.class)
@ContextConfiguration(locations="classpath:beans.xml")
public class MembershipTest {

  @Autowired
  UserService service;

  @Test
  public void test() {
```

```
            UserVO user = this.service.getUser("jimin");
            System.out.println(user);
            assertEquals("한지민", user.getName());
        }

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();
        session.insert("insert", user);
        session.commit();
        System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" + user.getName());
      } catch (IOException e) {
        e.printStackTrace();
      }
    }

    @Override
    public List<UserVO> readAll() {
      Reader rd = null;
      SqlSession session = null;
      List<UserVO> userList = null;
      try {
        rd = Resources.getResourceAsReader("mybatis-config.xml");
        session = new SqlSessionFactoryBuilder().build(rd).openSession();
        userList = session.selectList("selectAll");
      } catch (IOException e) {
        e.printStackTrace();
      }
      return userList;
    }

  3)MembershipTest.java
```

```
    @Autowired
    UserService service;

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

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)MembershipTest.java

```
    @Autowired
    UserService service;

    @Test
    public void test() {
        UserVO user = this.service.getUser("jimin");
        System.out.println(user);
```

```
1447          assertEquals("한지민", user.getName());
1448       }
1449       @Ignore @Test
1450       public void test1() {
1451          this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1452          for(UserVO user : this.service.getUserList()){
1453             System.out.println(user);
1454          }
1455       }
1456
1457       @Test
1458       public void test2() {
1459          service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1460          UserVO user = service.getUser("dooly");
1461          System.out.println(user);
1462       }
1463
1464    5. 사용자 정보 삭제 test
1465       1)mybatis-mapper.xml
1466
1467          <delete id="delete" parameterType="String">
1468             DELETE FROM USERS WHERE  userid = #{id}
1469          </delete>
1470
1471       2)UserDaoImplMyBatis.java
1472
1473       @Override
1474       public void delete(String id) {
1475          Reader rd = null;
1476          SqlSession session = null;
1477          UserVO userVO = null;
1478          try {
1479             rd = Resources.getResourceAsReader("mybatis-config.xml");
1480             session = new SqlSessionFactoryBuilder().build(rd).openSession();
1481             session.delete("delete", id);
1482             session.commit();
1483             System.out.println("삭제된 Record with ID = " + id );
1484          } catch (IOException e) {
1485             e.printStackTrace();
1486          }
1487       }
1488
1489       3)MembershipTest.java
1490
1491       @Autowired
1492       UserService service;
1493
1494       @Test
1495       public void test() {
1496          UserVO user = this.service.getUser("jimin");
1497          System.out.println(user);
1498          assertEquals("한지민", user.getName());
```

```
1499        }
1500        @Ignore @Test
1501        public void test1() {
1502           this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1503           for(UserVO user : this.service.getUserList()){
1504              System.out.println(user);
1505           }
1506        }
1507
1508        @Ignore @Test
1509        public void test2() {
1510           service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1511           UserVO user = service.getUser("dooly");
1512           System.out.println(user);
1513        }
1514
1515        @Test
1516        public void test3() {
1517           //사용자 정보 삭제 test
1518           service.deleteUser("dooly");
1519           for(UserVO user : service.getUserList()){
1520              System.out.println(user);
1521           }
1522        }
1523
1524
1525  ----------------------------------------------------
1526  Task6. Example of Spring JdbcTemplate
1527  1. Create Table
1528     create table employee(
1529        id number(10),
1530        name varchar2(100),
1531        salary number(10)
1532     );
1533
1534  2. Employee.java
1535     package com.example;
1536
1537     public class Employee {
1538        private int id;
1539        private String name;
1540        private float salary;
1541        //no-arg and parameterized constructors
1542        //getters and setters
1543     }
1544
1545  3. EmployeeDao.java
1546     package com.example;
1547     import org.springframework.jdbc.core.JdbcTemplate;
1548
1549     public class EmployeeDao {
1550        private JdbcTemplate jdbcTemplate;
```

```
1551
1552        public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
1553            this.jdbcTemplate = jdbcTemplate;
1554        }
1555
1556        public int saveEmployee(Employee e){
1557            String query="insert into employee values(
1558            '"+e.getId()+"','"+e.getName()+"','"+e.getSalary()+"')";
1559            return jdbcTemplate.update(query);
1560        }
1561        public int updateEmployee(Employee e){
1562            String query="update employee set
1563            name='"+e.getName()+"',salary='"+e.getSalary()+"' where id='"+e.getId()+"' ";
1564            return jdbcTemplate.update(query);
1565        }
1566        public int deleteEmployee(Employee e){
1567            String query="delete from employee where id='"+e.getId()+"' ";
1568            return jdbcTemplate.update(query);
1569        }
1570    }
1571
1572  4. applicationContext.xml
1573    <?xml version="1.0" encoding="UTF-8"?>
1574    <beans
1575        xmlns="http://www.springframework.org/schema/beans"
1576        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1577        xmlns:p="http://www.springframework.org/schema/p"
1578        xsi:schemaLocation="http://www.springframework.org/schema/beans
1579            http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
1580
1581        <bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
1582            <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
1583            <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />
1584            <property name="username" value="scott" />
1585            <property name="password" value="tiger" />
1586        </bean>
1587
1588        <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
1589            <property name="dataSource" ref="ds" />
1590        </bean>
1591
1592        <bean id="edao" class="com.example.EmployeeDao">
1593            <property name="jdbcTemplate" ref="jdbcTemplate" />
1594        </bean>
1595
1596    </beans>
1597
1598  5. Test.java
1599    package com.example;
1600
1601    import org.springframework.context.ApplicationContext;
1602    import org.springframework.context.support.ClassPathXmlApplicationContext;
```

```
1603    public class Test {
1604
1605        public static void main(String[] args) {
1606            ApplicationContext ctx=new ClassPathXmlApplicationContext("applicationContext.xml");
1607
1608            EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");
1609            int status=dao.saveEmployee(new Employee(102,"Amit",35000));
1610            System.out.println(status);
1611
1612            /*int status=dao.updateEmployee(new Employee(102,"Sonoo",15000));
1613                System.out.println(status);
1614            */
1615
1616            /*Employee e=new Employee();
1617            e.setId(102);
1618            int status=dao.deleteEmployee(e);
1619            System.out.println(status);*/
1620        }
1621    }
1622
1623
1624  ----------------------------------------------------
1625  Task7. Example of PreparedStatement in Spring JdbcTemplate
1626  1. Create Table
1627     create table employee(
1628        id number(10),
1629        name varchar2(100),
1630        salary number(10)
1631     );
1632
1633  2. Employee.java
1634     package com.example;
1635
1636     public class Employee {
1637        private int id;
1638        private String name;
1639        private float salary;
1640        //no-arg and parameterized constructors
1641        //getters and setters
1642     }
1643
1644  3. EmployeeDao.java
1645     package com.example;
1646     import java.sql.PreparedStatement;
1647     import java.sql.SQLException;
1648
1649     import org.springframework.dao.DataAccessException;
1650     import org.springframework.jdbc.core.JdbcTemplate;
1651     import org.springframework.jdbc.core.PreparedStatementCallback;
1652
1653     public class EmployeeDao {
1654        private JdbcTemplate jdbcTemplate;
```

```
1655
1656        public void setJdbcTemplate(JdbcTemplate jdbcTemplate) {
1657            this.jdbcTemplate = jdbcTemplate;
1658        }
1659
1660        public Boolean saveEmployeeByPreparedStatement(final Employee e){
1661            String query="insert into employee values(?,?,?)";
1662            return jdbcTemplate.execute(query,new PreparedStatementCallback<Boolean>(){
1663                @Override
1664                public Boolean doInPreparedStatement(PreparedStatement ps)
1665                        throws SQLException, DataAccessException {
1666
1667                    ps.setInt(1,e.getId());
1668                    ps.setString(2,e.getName());
1669                    ps.setFloat(3,e.getSalary());
1670
1671                    return ps.execute();
1672
1673                }
1674            });
1675        }
1676    }
1677
1678  4. applicationContext.xml
1679      <?xml version="1.0" encoding="UTF-8"?>
1680      <beans
1681          xmlns="http://www.springframework.org/schema/beans"
1682          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1683          xmlns:p="http://www.springframework.org/schema/p"
1684          xsi:schemaLocation="http://www.springframework.org/schema/beans
1685              http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
1686
1687          <bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
1688              <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
1689              <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />
1690              <property name="username" value="scott" />
1691              <property name="password" value="tiger" />
1692          </bean>
1693
1694          <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
1695              <property name="dataSource" ref="ds" />
1696          </bean>
1697
1698          <bean id="edao" class="com.example.EmployeeDao">
1699              <property name="jdbcTemplate" ref="jdbcTemplate" />
1700          </bean>
1701      </beans>
1702
1703  5. Test.java
1704      package com.example;
1705
1706      import org.springframework.context.ApplicationContext;
```

```
1707    import org.springframework.context.support.ClassPathXmlApplicationContext;
1708    public class Test {
1709
1710        public static void main(String[] args) {
1711            ApplicationContext ctx=new ClassPathXmlApplicationContext("applicationContext.xml");
1712
1713            EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");
1714            dao.saveEmployeeByPreparedStatement(new Employee(108,"Amit",35000));
1715        }
1716    }
1717
1718
1719  ----------------------------------------------------
1720  Task8. ResultSetExtractor Example | Fetching Records by Spring JdbcTemplate
1721  1. Create Table
1722     create table employee(
1723        id number(10),
1724        name varchar2(100),
1725        salary number(10)
1726     );
1727
1728  2. Employee.java
1729     package com.example;
1730
1731     public class Employee {
1732        private int id;
1733        private String name;
1734        private float salary;
1735        //no-arg and parameterized constructors
1736        //getters and setters
1737        public String toString(){
1738           return id+" "+name+" "+salary;
1739        }
1740     }
1741
1742  3. EmployeeDao.java
1743     package com.example;
1744     import java.sql.ResultSet;
1745     import java.sql.SQLException;
1746     import java.util.ArrayList;
1747     import java.util.List;
1748     import org.springframework.dao.DataAccessException;
1749     import org.springframework.jdbc.core.JdbcTemplate;
1750     import org.springframework.jdbc.core.ResultSetExtractor;
1751
1752     public class EmployeeDao {
1753        private JdbcTemplate template;
1754
1755        public void setTemplate(JdbcTemplate template) {
1756           this.template = template;
1757        }
1758
```

```
1759        public List<Employee> getAllEmployees(){
1760         return template.query("select * from employee",new ResultSetExtractor<List<Employee>>(){
1761           @Override
1762            public List<Employee> extractData(ResultSet rs) throws SQLException,
1763                DataAccessException {
1764
1765              List<Employee> list=new ArrayList<Employee>();
1766              while(rs.next()){
1767                 Employee e=new Employee();
1768                 e.setId(rs.getInt(1));
1769                 e.setName(rs.getString(2));
1770                 e.setSalary(rs.getInt(3));
1771                 list.add(e);
1772              }
1773              return list;
1774           }
1775           });
1776          }
1777        }
1778     }
1779
1780  4. applicationContext.xml
1781     <?xml version="1.0" encoding="UTF-8"?>
1782     <beans
1783        xmlns="http://www.springframework.org/schema/beans"
1784        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1785        xmlns:p="http://www.springframework.org/schema/p"
1786        xsi:schemaLocation="http://www.springframework.org/schema/beans
1787           http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
1788
1789        <bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
1790           <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
1791           <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />
1792           <property name="username" value="scott" />
1793           <property name="password" value="tiger" />
1794        </bean>
1795
1796        <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
1797           <property name="dataSource" ref="ds" />
1798        </bean>
1799
1800        <bean id="edao" class="com.example.EmployeeDao">
1801           <property name="jdbcTemplate" ref="jdbcTemplate" />
1802        </bean>
1803
1804     </beans>
1805
1806  5. Test.java
1807     package com.example;
1808
1809     import java.util.List;
1810
```

```
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class Test {

    public static void main(String[] args) {
        ApplicationContext ctx=new ClassPathXmlApplicationContext("applicationContext.xml");
        EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");
        List<Employee> list=dao.getAllEmployees();

        for(Employee e:list)
            System.out.println(e);

    }
}


-------------------------------------------------
Task9. RowMapper Example | Fetching records by Spring JdbcTemplate
1. Create Table
  create table employee(
    id number(10),
    name varchar2(100),
    salary number(10)
  );

2. Employee.java
  package com.example;

  public class Employee {
    private int id;
    private String name;
    private float salary;
    //no-arg and parameterized constructors
    //getters and setters
    public String toString(){
      return id+" "+name+" "+salary;
    }
  }

3. EmployeeDao.java
  package com.example;
  import java.sql.ResultSet;
  import java.sql.SQLException;
  import java.util.ArrayList;
  import java.util.List;
  import org.springframework.dao.DataAccessException;
  import org.springframework.jdbc.core.JdbcTemplate;
  import org.springframework.jdbc.core.ResultSetExtractor;

  public class EmployeeDao {
    private JdbcTemplate template;

```

```
    public void setTemplate(JdbcTemplate template) {
        this.template = template;
    }

    public List<Employee> getAllEmployees(){
        return template.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;
            }
        });
    }
}
```

4. applicationContext.xml

```
<?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:p="http://www.springframework.org/schema/p"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

    <bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
        <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
        <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />
        <property name="username" value="scott" />
        <property name="password" value="tiger" />
    </bean>

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

    <bean id="edao" class="com.example.EmployeeDao">
        <property name="jdbcTemplate" ref="jdbcTemplate"></property>
    </bean>

</beans>
```

5. Test.java

```
package com.example;

import java.util.List;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class Test {
```

```
1915
1916        public static void main(String[] args) {
1917            ApplicationContext ctx=new ClassPathXmlApplicationContext("applicationContext.xml");
1918            EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");
1919            List<Employee> list=dao.getAllEmployeesRowMapper();
1920
1921            for(Employee e:list)
1922                System.out.println(e);
1923
1924        }
1925    }
1926
1927
1928
1929  ----------------------------------------------------
1930  Task10. Spring NamedParameterJdbcTemplate Example
1931  1. Create Table
1932     create table employee(
1933        id number(10),
1934        name varchar2(100),
1935        salary number(10)
1936     );
1937
1938  2. Employee.java
1939     package com.example;
1940
1941     public class Employee {
1942        private int id;
1943        private String name;
1944        private float salary;
1945        //no-arg and parameterized constructors
1946        //getters and setters
1947     }
1948
1949  3. EmployeeDao.java
1950     package com.example;
1951
1952     import java.sql.PreparedStatement;
1953     import java.sql.SQLException;
1954     import org.springframework.dao.DataAccessException;
1955     import org.springframework.jdbc.core.PreparedStatementCallback;
1956     import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
1957     import java.util.*;
1958
1959     public class EmpDao {
1960        NamedParameterJdbcTemplate template;
1961
1962        public EmpDao(NamedParameterJdbcTemplate template) {
1963            this.template = template;
1964        }
1965        public  void save (Emp e){
1966            String query="insert into employee values (:id,:name,:salary)";
```

```
1967
1968            Map<String,Object> map=new HashMap<String,Object>();
1969            map.put("id",e.getId());
1970            map.put("name",e.getName());
1971            map.put("salary",e.getSalary());
1972
1973            template.execute(query,map,new PreparedStatementCallback() {
1974                @Override
1975                public Object doInPreparedStatement(PreparedStatement ps)
1976                        throws SQLException, DataAccessException {
1977                    return ps.executeUpdate();
1978                }
1979            });
1980        }
1981    }
1982
1983  4. applicationContext.xml
1984    <?xml version="1.0" encoding="UTF-8"?>
1985    <beans
1986        xmlns="http://www.springframework.org/schema/beans"
1987        xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1988        xmlns:p="http://www.springframework.org/schema/p"
1989        xsi:schemaLocation="http://www.springframework.org/schema/beans
1990            http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
1991
1992        <bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
1993            <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
1994            <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />
1995            <property name="username" value="scott" />
1996            <property name="password" value="tiger" />
1997        </bean>
1998
1999        <bean id="jtemplate"
        class="org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate">
2000            <constructor-arg ref="ds" />
2001        </bean>
2002
2003        <bean id="edao" class="com.example.EmpDao">
2004            <constructor-arg>
2005                <ref bean="jtemplate"/>
2006            </constructor-arg>
2007        </bean>
2008
2009    </beans>
2010
2011  5. Test.java
2012    package com.example;
2013
2014    import org.springframework.beans.factory.BeanFactory;
2015    import org.springframework.beans.factory.xml.XmlBeanFactory;
2016    import org.springframework.core.io.ClassPathResource;
2017    import org.springframework.core.io.Resource;
```

```
2018
2019    public class SimpleTest {
2020       public static void main(String[] args) {
2021
2022          Resource r=new ClassPathResource("applicationContext.xml");
2023          BeanFactory factory=new XmlBeanFactory(r);
2024
2025          EmpDao dao=(EmpDao)factory.getBean("edao");
2026          dao.save(new Emp(23,"sonoo",50000));
2027
2028       }
2029    }
2030
2031
2032 ----------------------------------------------------
2033 Task11. Spring SimpleJdbcTemplate Example
2034 1. Create Table
2035    create table employee(
2036       id number(10),
2037       name varchar2(100),
2038       salary number(10)
2039    );
2040
2041 2. Employee.java
2042    package com.example;
2043
2044    public class Employee {
2045       private int id;
2046       private String name;
2047       private float salary;
2048       //no-arg and parameterized constructors
2049       //getters and setters
2050    }
2051
2052 3. EmployeeDao.java
2053    package com.example;
2054
2055    import org.springframework.jdbc.core.simple.SimpleJdbcTemplate;
2056    public class EmpDao {
2057       SimpleJdbcTemplate template;
2058
2059       public EmpDao(SimpleJdbcTemplate template) {
2060          this.template = template;
2061       }
2062       public int update (Emp e){
2063          String query="update employee set name=? where id=?";
2064          return template.update(query,e.getName(),e.getId());
2065
2066          //String query="update employee set name=?,salary=? where id=?";
2067          //return template.update(query,e.getName(),e.getSalary(),e.getId());
2068       }
2069    }
```

4. applicationContext.xml

```
<?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:p="http://www.springframework.org/schema/p"
   xsi:schemaLocation="http://www.springframework.org/schema/beans
      http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

   <bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
      <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
      <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />
      <property name="username" value="scott" />
      <property name="password" value="tiger" />
   </bean>

   <bean id="jtemplate" class="org.springframework.jdbc.core.simple.SimpleJdbcTemplate">
      <constructor-arg ref="ds" />
   </bean>

   <bean id="edao" class="com.example.EmpDao">
      <constructor-arg>
         <ref bean="jtemplate"/>
      </constructor-arg>
   </bean>

</beans>
```

5. Test.java

```
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");
      int status=dao.update(new Emp(23,"Tarun",35000));
      System.out.println(status);
   }
}
```