1. Spring JDBC Architecture
  1)개요
    -대부분의 중/대규모 웹 어플리케이션은 효율적인 개발 및 유지 보수를 위하여 계층화(Layering)하여 개발하는 것이 원칙이다.
    -사용자관리 프로젝트 아키텍처에서 기본적으로 가지는 계층은 Presentation Layer, Service Layer, Data Access Layer 3계층과 모든 계층에서 사용되는 Domain Model Class로 구성되어 있다.
    -각각의 계층은 계층마다 독립적으로 분리하여 구현하는 것이 가능해야 하며, 각 계층에서 담당해야 할 기능들이 있다.

  2)Architecture 개요
    Presentation <---> Service <---> Data Access
      Layer Layer Layer
    <----------Domain Model Class ---------->

    -위의 3가지 계층은 독립적으로 분리할 수 있도록 구현해야 하며, 일반적으로 각 계층 사이에서는 interface를 이용하여 통신하는 것이 일반적이다.

2. Presentation Layer
  1)Browser상의 웹 클라이언트의 요청 및 응답을 처리
  2)상위계층(서비스 계층, 데이터 액세스 계층)에서 발생하는 Exception에 대한 처리
  3)최종 UI에서 표현해야 할 도메인 모델을 사용
  4)최종 UI에서 입력한 데이터에 대한 유효성 검증(Validation) 기능을 제공
  5)비즈니스 로직과 최종 UI를 분리하기 위한 컨트롤러 기능을 제공
  6)@Controller 어노테이션을 사용하여 작성된 Controller 클래스가 이 계층에 속함

3. Service Layer
  1)어플리케이션 비즈니스 로직 처리와 비즈니스와 관련된 도메인 모델의 적합성 검증
  2)Transaction 처리
  3)Presentation Layer와 Data Access Layer 사이를 연결하는 역할로서 두 계층이 직접적으로 통신하지 않게 하여 어플리케이션의 유연성을 증가
  4)다른 계층들과 통신하기 위한 인터페이스 제공
  5)Service 인터페이스와 @Service 어노테이션을 사용하여 작성된 Service 구현 클래스가 이 계층에 속함.

4. Data Access Layer
  1)영구 저장소(관계형 데이터베이스)의 데이터를 조작하는 데이터 액세스 로직을 객체화
  2)영구 저장소의 데이터를 조회, 등록, 수정, 삭제함
  3)ORM(Object Relational Mapping) 프레임워크(MyBatis, Hibernate)를 주로 사용하는 계층
  4)DAO 인터페이스와 @Repository 어노테이션을 사용하여 작성된 DAO 구현 클래스가 이 계층에 속함.

5. Domain Model Class
  1)관계형 데이터 베이스의 엔티티와 비슷한 개념을 가지는 것으로 실제 VO(Value Object) 혹은 DTO(Data Transfer Object) 객체에 해당
  2)도메인 모델 클래스는 3개의 계층 전체에 걸쳐 사용
  3)private으로 선언된 멤버변수가 있고, 그 변수에 대한 getter와 setter 메소드를 가진 클래스를 말함.

6. 데이터 액세스 공통 개념
  1)DAO(Data Access Object) Pattern
    -데이터 액세스 계층은 DAO 패턴을 적용하여 비즈니스 로직과 데이터 액세스 로직을 분리하는 것이 원칙이다.
    -데이터베이스 접속과 SQL 발행 같은 데이터 액세스 처리를 DAO라고 불리는 오브젝트로 분리하는 패턴이다.
    -비즈니스 로직이 없거나 단순하면 DAO와 서비스 계층을 통합할 수도 있지만, 의미 있는 비즈니스 로직을 가진 엔터프라

이즈 어플리케이션이라면 데이터 액세스 계층을 DAO 패턴으로 분리해야 한다.
-DAO 패턴은 서비스계층에 영향을 주지 않고 데이터 액세스 기술을 변경할 수 있는 장점을 가지고 있다.
-비즈니스 로직 -> Dao 인터페이스 -> XxxDao(CRUD구현) -> Database
-Dao클래스에 데이터 액세스 처리를 기술하겠지만, 그 처리를 구현하는 Java기술은 여러 가지다.
--Dao Interface -> XxxDao(CRUD) -> JDBC/Hibernate/MyBatis(iBATIS)/JPA/JDO등등등 -> Database
--Spring에서는 새로운 데이터 액세스 기술을 제공하는 것이 아니라 기존의 5가지 방법 기술들을 좀 더 쉽게 만드는 기능을 제공한다.
--즉 JDBC는 Spring JDBC로, Hibernate는 Hibernate 연계로, JPA는 JPA 연계로, MyBatis는 MyBatis 연계로, JDO는 JDO 연계이다.
-Spring의 기능을 이용해서 얻을 수 있는 장점을 아래 3가지 이다.
--데이터 액세스 처리를 간결하게 기술할 수 있다.
--스프링이 제공하는 범용적이고 체계적인 데이터 액세스 예외를 이용할 수 있다.
--스프링의 트랙잭션 기능을 이용할 수 있다.
--여기서는 5가지 기술 중 Spring JDBC를 다룬다.

2)Connection Pooling을 지원하는 DataSource
-Connection Pooling은 미리 정해진 갯수만큼의 DB Connection을 Pool에 준비해두고, 어플리케이션이 요청할 때마다 Pool에서 꺼내서 하나씩 할당해주고 다시 돌려받아서 Pool에 넣는 식의 기법이다.
-다중 사용자를 갖는 엔터프라이즈 시스템에서라면 반드시 DB Connection Pooling 기능을 지원하는 DataSource를 사용해야 한다.
-Spring에서는 DataSource를 공유 가능한 Spring Bean으로 등록해 주어 사용할 수 있도록 해준다.

7. DataSource 구현 클래스의 종류
1)테스트 환경을 위한 DataSource
-SimpleDriverDataSource
--Spring이 제공하는 가장 단순한 DataSource 구현 클래스
--getConnection()을 호출할 때마다 매번 DB Connection을 새로 만들고 따로 Pool을 관리하지 않으므로 단순한 테스트용으로만 사용해야 한다.

-SingleConnectionDriverDataSource
--순차적으로 진행되는 통합 테스트에서는 사용 가능하다.
--매번 DB Connection을 생성하지 않기 때문에 SimpleDriverDataSource보다 빠르게 동작한다.

2)OpenSource DataSource
-Apache Commons DBCP
--가장 유명한 오픈소스 DB Conneciton Pool Library이다.
--Apache의 Commons Project(http://commons.apache.org/dbcp)
-c3p0 JDBC/DataSource Resource Pool
--c3p0는 JDBC 3.0 스펙을 준수하는 Connection과 Statement Pool을 제공하는 라이브러리이다.
--http://www.mchange.com/projects/c3p0/
--두 가지 모두 수정자(setter0 메소드를 제공하므로 Spring Bean으로 등록해서 사용하기 편리.

8. Spring JDBC
1)JDBC란?
-모든 Java의 Data Access 기술의 근간
-Entity Class와 Annotation을 이용하는 최신 ORM 기술도 내부적으로는 DB와의 연동을 위해 JDBC를 이용
-안정적이고 유연한 기술이지만, Low level 기술로 인식되고 있다.
-간단한 SQL을 실행하는데도 중복된 코드가 반복적으로 사용되며, DB에 따라 일관성 없는 정보를 가진 채 Checked Exception으로 처리한다.
-장점
--대부분의 개발자가 잘 알고 있는 친숙한 데이터 액세스 기술로 별도의 학습 없이 개발이 가능

-단점
--Connection과 같은 공유 리소스를 제대로 릴리즈 해주지 않으면 시스템의 자원이 바닥나는 버그 발생.

2)Spring JDBC?
-JDBC의 장점과 단순성을 그대로 유지하면서도 기존 JDBC의 단점을 극복
-간결한 형태의 API 사용법을 제공
-JDBC API에서 지원되지 않는 편리한 기능 제공
-반복적으로 해야 하는 많은 작업들을 대신 해줌.
-Spring JDBC를 사용할 때는 실행할 SQL과 바인딩 할 파라미터를 넘겨주거나, 쿼리의 실행 결과를 어떤 객체에서 넘겨 받을지를 지정하는 것만 하면 된다.
-Spring JDBC를 사용하려면 먼저, DB Connection을 가져오는 DataSource를 Bean으로 등록해야 한다.

3)개발자가 JDBC방식으로 연결시의 문제점들
-직접 개발자가 JDBC를 사용하면 소스 코드가 너무 길어지고 또한 커넥션이나 PreparedStatement를 얻고 나면 반드시 연결 해제를 처리해야 하지만 깜빡 잊어버리는 개발자도 있을 수 있다.
-그래서 연결이 해제되지 않으면 데이터베이스의 리소스 고갈이나 메모리 누수의 원인이 되어 최악의 경우에는 시스템이 정지할 가능성도 있다.
-데이터 액세스 오류시 오류 원인을 특정하고 싶을 때는 SQLException의 오류 코드를 가져와 값을 조사할 필요가 있다.
-더욱이 오류 코드는 데이터베이스 제품마다 값이 다르므로 데이터베이스 제품이 바뀌면 다시 수정해야만 한다.
-또한 SQLException은 컴파일 시 예외 처리 유무를 검사하므로 소스 코드 상에서 반드시 catch 문을 기술해야만 한다.

4)Spring JDBC가 해주는 작업들
-Connection 열기와 닫기
--Connection과 관련된 모든 작업을 Spring JDBC가 필요한 시점에서 알아서 진행한다.
--진행 중에 예외가 발생했을 때도 열린 모든 Connection 객체를 닫아준다.
-Statement 준비와 닫기
--SQL 정보가 담긴 Statement 또는 PreparedStatement를 생성하고 필요한 준비 작업을 한다.
--Statement도 Connection과 마찬가지로 사용이 끝나면 Spring JDBC가 알아서 닫아준다.
-Statement 실행
--SQL이 담긴 Statement를 실행
--Statement의 실행결과를 다양한 형태로 가져올 수 있다.
-ResultSet Loop 처리
--ResultSet에 담긴 쿼리 실행 결과가 한 건 이상이면 ResultSet 루프를 만들어서 반복한다.
-Exception 처리와 반환
--JDBC 작업 중 발생하는 모든 예외는 Spring JDBC 예외 변환기가 처리한다.
--Checked Exception인 SQLException을 Runtime Exception인 DataAccessException 타입으로 변환
-Transaction 처리
--Transaction과 관련된 모든 작업에 대해서는 신경쓰지 않아도 된다.

5)Spring JDBC의 JdbcTemplate Class
-Spring JDBC가 제공하는 클래스 중 하나
-JDBC의 모든 기능을 최대한 활용할 수 있는 유연성을 제공하는 클래스
-실행, 조회, 배치의 3가지 작업 제공
--실행 : Insert나 Update같이 DB의 데이터에 변경이 일어나는 쿼리를 수행하는 작업
--조회 : Select를 이용해 데이터를 조회하는 작업
--배치 : 여러 개의 쿼리를 한번에 수행해야 하는 작업

6)JdbcTemplate class 생성
-JdbcTemplate은 DataSource를 파라미터로 받아서 아래와 같이 생성한다.
JdbcTemplate template = new JdbcTemplate(dataSource);

-DataSource는 보통 Bean으로 등록해서 사용하므로 JdbcTemplate이 필요한 DAO class에서 DataSource Bean을 DI 받아서 JdbcTemplate을 생성할 때 인자로 넘겨주면 된다.
-JdbcTemplate은 멀티스레드 환경에서도 안전하게 공유해서 쓸 수 있기 때문에 DAO class의 인스턴스 변수에 저장해 두고 사용할 수 있다.
-생성 예

```
public class UserDAOJdbc{
  JdbcTemplate jdbcTemplate;

  @Autowired
  public void setDataSource(DataSource dataSource){
    jdbcTemplate = new JdbcTemplate(dataSource);
  }
}
```

7)JdbcTemplate의 Update() 메소드
-INSERT, UPDATE, DELETE와 같은 SQL을 실행할 때 사용.
  int update(String sql, [SQL 파라미터])
-이 메소드를 호출할 때는 SQL과 함께 바인딩 할 파라미터는 Object 타입 가변인자(Object ... args)를 사용할 수 있다.
-이 메소드의 리턴값은 SQL 실행으로 영향받은 레코드의 갯수이다.
-사용 예

```
public int update(User user){
  StringBuffer updateQuery = new StringBuffer();
  updateQuery.append("UPDATE USERS SET ");
  updateQuery.append("password=?, name=? ");
  updateQuery.append("WHERE id=? ");

  int result = this.jdbcTemplate.update(updateQuery.toString(),
                          user.getName(), user.getPassword(), user.getId());
  return result;
}
```

8)JdbcTemplate의 queryForObject() 메소드
-SELECT SQL을 실행하여 하나의 Row를 가져올 때 사용.
  <T> T queryForObject(String sql, [SQL 파라미터], RowMapper<T> rm)
-SQL 실행 결과는 여러 개의 칼럼을 가진 하나의 Row
-T는 VO 객체의 타입에 해당
-SQL 실행 결과로 돌아온 여러 개의 Column을 가진 한 개의 Row를 RowMapper 콜백을 이용해 VO 객체로 매핑한다.
-사용 예

```
public User findUser(String id){
  return this.jdbcTemplate.queryForObject("SELECT * FROM users WHERE id=?",
    new Object [] {id},
    new RowMapper<User>(){
      public User mapRow(ResultSet rs, int rowNum) throws SQLException{
        User user = new User();
        user.setId(rs.getString("id"));
        user.setName(rs.getString("name"));
        user.setPassword(rs.getString("password"));
        return user;
      }
```

```
            }
        }
```

9)JdbcTemplate 클래스의 query() 메소드
-SELECT SQL을 실행하여 여러 개의 Row를 가져올 때 사용.

<T> List<T> query(String sql, [SQL 파라미터], RowMapper<T> rm)

-SQL 실행 결과로 돌아온 여러 개의 Column을 가진 여러 개의 Row를 RowMapper 콜백을 이용해 VO 객체로 매핑해준다.
-결과 값은 매핑 한 VO 객체를 포함하고 있는 List 형태로 받는다.
-List의 각 요소가 하나의 Row에 해당한다.

9. Spring JDBC 환경설정

1)Oracle Jdbc Driver 라이브러리 검색 및 설치

****

※Oracle의 경우 어떤 드라이버를 pom.xml에 넣어도 에러가 난다.
원래는 Oracle 12C인 경우 Maven Repository에서 'oracle ojdbc8'으로, Oracle 11g인 경우는 'oracle ojdbc6'로 검색해야 한다.
1)'oracle ojdbc7'으로 검색시 12.1.0.2

```
<!-- https://mvnrepository.com/artifact/com.github.noraui/ojdbc7 -->
<dependency>
    <groupId>com.github.noraui</groupId>
    <artifactId>ojdbc7</artifactId>
    <version>12.1.0.2</version>
</dependency>
```

2)'oracle ojdbc6'으로 검색시 11.1.0.7.0

```
<!-- https://mvnrepository.com/artifact/com.oracle/ojdbc6 -->
<dependency>
    <groupId>com.oracle</groupId>
    <artifactId>ojdbc6</artifactId>
    <version>11.1.0.7.0</version>
    <scope>test</scope>
</dependency>
```

하지만 어떤 버전도 Maven에서 에러가 난다. http://suyou.tistory.com/68 참조.
메이븐에서 Oracle 드라이버를 찾지 못하는 것은 아마도 저작권 문제로 보인다.
그래서 Oracle 사이트에서 직접 드라이버를 다운로드 받아서 Maven을 이용해서 Maven Local Repository에 인스톨을 하고
인스톨된 버전으로 pom.xml에 디펜던시 설정을 해야 한다.

① 오라클 홈페이지에서 Orcle 12C jdbc드라이버를 다운로드 받는다. -->ojdbc8.jar
② 메이븐 인스톨러를 이용해서 메이븐 레포지토리에 설치한다.
mvn install:install-file -Dfile="파일이름(위치까지)" -DgroupId=그룹아이디 -DartifactId=파일이름 -Dversion=버전 -Dpackaging=jar
위에명령을 cmd에서 실행한다.
자기버전에 맞게 해당항목을 변경한다음 실행한다.

```
C:\Windows\system32>mvn install:install-file -Dfile="C:\temp\ojdbc8.jar"
-DgroupId=com.oracle -DartifactId=ojdbc8 -Dversion=12.2 -Dpackaging=jar
```

```
        [INFO] Scanning for projects...
        [INFO]
        [INFO] ------------------< org.apache.maven:standalone-pom >------------------
        [INFO] Building Maven Stub Project (No POM) 1
        [INFO] --------------------------------[ pom ]---------------------------------
        [INFO]
        [INFO] --- maven-install-plugin:2.4:install-file (default-cli) @ standalone-pom
        ---
        [INFO] Installing C:\temp\ojdbc8.jar to C:\Users\user\.m2\repository\com\oracle
        \ojdbc8\12.2\ojdbc8-12.2.jar
        [INFO] Installing C:\Users\user\AppData\Local\Temp\mvninstall1974190192777781278
        .pom to C:\Users\user\.m2\repository\com\oracle\ojdbc8\12.2\ojdbc8-12.2.pom
        [INFO] ------------------------------------------------------------------------
        [INFO] BUILD SUCCESS
        [INFO] ------------------------------------------------------------------------
        [INFO] Total time:  0.383 s
        [INFO] Finished at: 2018-12-04T12:39:07+09:00
        [INFO] ------------------------------------------------------------------------

        인스톨 명령을 실행하면 메이븐 depository에 해당 드라이버가 설치된다.
        위에서는 C:\Users\webnbiz01\.m2\repository\com\oracle\ojdbc8\12.2 에 설치된 것이다.
        해당 디렉토리로 이동하면 jar 파일과 pom 파일이 있다.
        pom파일의 groupId, artifactId, version을 pom.xml에 디팬던시로 설정하면 된다.

    ③ pom.xml에 디펜던시를 설정한다.
        <dependency>
         <groupId>com.oracle</groupId>
         <artifactId>ojdbc8</artifactId>
         <version>12.2</version>
        </dependency>

        pom.xml에 추가한다.
        이후 pom.xml clean후 install 한다.
**********************************************************************************

  2)Spring JDBC 설치
    -Maven Repository에서 'Spring jdbc'라고 검색
    -JdbcTemplate를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.

        <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
        <dependency>
            <groupId>org.springframework</groupId>
            <artifactId>spring-jdbc</artifactId>
            <version>4.3.13.RELEASE</version>
        </dependency>


10. Lab
  1)SpringJdbcDemo project 생성
    -New > Java Project >
    -Project name : SpringJdbcDemo > Finish

  2)com.example Package 생성
    -/src > right-click > New > Package
```

297 -Name : com.example > Finish
298
299 3)config 폴더 생성
300 -SpringJdbcDemo project > right-click > Build Path > Coinfigure Build Path
301 -Source Tab > Add Folder > Select SpringJdbcDemo project > Click [Create New Folder] button
302 -Folder name : config > Finish > OK
303
304 4)config/dbinfo.properties 파일 생성
305 -config > right-click > New > File
306 -File name : dbinfo.properties > Finish
307
308 db.driverClass=oracle.jdbc.driver.OracleDriver
309 db.url=jdbc:oracle:thin:@localhost:1521:XE
310 db.username=scott
311 db.password=tiger
312
313 5)/src/com.example.UserClient.java 생성
314 -/src > com.example > right-click > New > Class
315 -Name : UserClient
316
317 public class UserClient{
318 public static void main(String [] args){
319
320 }
321 }
322
323 6)Maven Project로 전환
324 -SpringJdbcDemo Project > right-click > Configure > Convert to Maven Project
325 -Finish
326
327 7)Spring Project로 전환
328 -SpringJdbcDemo Project > right-click > Spring Tools > Add Spring Project Nature
329
330 8)Oracle Jdbc Driver 라이브러리 검색 및 설치
331
332 9)Spring Context 설치
333 -Maven Repository 에서 'Spring Context'로 검색하여 디펜던시 추가하고 설치
334
335 <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
336 <dependency>
337 <groupId>org.springframework</groupId>
338 <artifactId>spring-context</artifactId>
339 <version>4.3.24.RELEASE</version>
340 </dependency>
341 -pom.xml에 붙여 넣고 Maven Install 하기
342
343 10)Spring JDBC 설치
344 -JdbcTemplate를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.
345
346 <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
347 <dependency>
348 <groupId>org.springframework</groupId>
349 <artifactId>spring-jdbc</artifactId>

```
            <version>4.3.24.RELEASE</version>
        </dependency>

    -pom.xml에 붙여 넣고 Maven Install 하기

11)Bean Configuration XML 작성
    -/src/config > right-click > New > Other > Spring > Spring Bean Configuration File
    -File name : beans.xml > Next
    -Check [beans - http://www.springframework.org/schema/beans]
    -Check [http://www.springframework.org/schema/beans/spring-beans-4.3.xsd]
    -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>

    -Namespace tab에서 context - http://www.springframework.org/schema/context check

        <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>

12)/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);
                }
            }
        }
```

13)Test
oracle.jdbc.driver.T4CConnection@51c8530f

11. 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)Class Diagram
Membership Class Diagram.uml 파일 참조

3)각 Class의 역할
-Presentation Layer
--UserController<Class>
---UI계층과 서비스 계층을 연결하는 역할을 하는 클래스
---JSP에서 UserController를 통해서 서비스 계층의 UserService를 사용하게 된다.
---Service 계층의 UserService 인터페이스를 구현하나 객체를 IoC 컨테이너가 주입해 준다.

-Service Layer
--UserService<Interface>
---서비스 계층에 속한 상위 인터페이스
--UserServiceImpl<Class>
---UserSerive 인터페이스를 구현한 클래스
---복잡한 업무 로직이 있을 경우에는 이 클래스에서 업무 로직을 구현하면 된다.
---데이터 액세스 계층의 userDao 인터페이스를 구현한 객체를 IoC 컨테이너가 주입해준다.

-Data Access Layer
--UserDao<Interface>
---데이터 엑세스 계층에 속한 상위 인터페이스
--UserDaoImplJDBC<Class> - Spring JDBC 구현
---UserDao 인터페이스를 구현한 클래스로 이 클래스에서는 데이터 액세스 로직을 구현하면 된다.
---Spring JDBC를 사용하는 경우에는 DataSource를 IoC 컨테이너가 주입해준다.
---MyBatis를 사용하는 경우에는 SqlSession을 IoC 컨테이너가 주입해준다.

4)New > Spring Legacy Project > Simple Spring Maven
Project name : Membership

5)/src/main/java > right-click > New > Package
Package name : com.example.vo

6)UserVO.java 생성

```
package com.example.vo;
```

```
457
458     public class UserVO {
459
460       private String userId;
461       private String name;
462       private String gender;
463       private String city;
464
465       public UserVO() {}
466
467       public UserVO(String userId, String name, String gender, String city) {
468         this.userId = userId;
469         this.name = name;
470         this.gender = gender;
471         this.city = city;
472       }
473
474       public String getUserId() {
475         return userId;
476       }
477
478       public void setUserId(String userId) {
479         this.userId = userId;
480       }
481
482       public String getName() {
483         return name;
484       }
485
486       public void setName(String name) {
487         this.name = name;
488       }
489
490       public String getGender() {
491         return gender;
492       }
493
494       public void setGender(String gender) {
495         this.gender = gender;
496       }
497
498       public String getCity() {
499         return city;
500       }
501
502       public void setCity(String city) {
503         this.city = city;
504       }
505
506       @Override
507       public String toString() {
508         return "User [userId=" + userId + ", name=" + name + ", gender="
509             + gender + ", city=" + city + "]";
510       }
```

```
511     }
512
513   7)/src/main/java > right-click > New > Package
514       Package name : com.example.service
515
516   8)UserService.java
517
518     package com.example.service;
519
520     import java.util.List;
521     import com.example.vo.UserVO;
522
523     public interface UserService {
524
525       void insertUser(UserVO user);
526
527       List<UserVO> getUserList();
528
529       void deleteUser(String id);
530
531       UserVO getUser(String id);
532
533       void updateUser(UserVO user);
534     }
535
536   9)UserServiceImpl.java
537       package com.example.service;
538
539       import java.util.List;
540       import com.example.vo.UserVO;
541
542       public class UserServiceImpl implements UserService {
543
544         @Override
545         public void insertUser(UserVO user) {
546           // TODO Auto-generated method stub
547
548         }
549
550         @Override
551         public List<UserVO> getUserList() {
552           // TODO Auto-generated method stub
553           return null;
554         }
555
556         @Override
557         public void deleteUser(String id) {
558           // TODO Auto-generated method stub
559
560         }
561
562         @Override
563         public UserVO getUser(String id) {
564           // TODO Auto-generated method stub
```

```
565           return null;
566         }
567
568         @Override
569         public void updateUser(UserVO user) {
570           // TODO Auto-generated method stub
571
572         }
573       }
574
575   10)/src/main/java > right-click > New > Package
576       Package name : com.example.dao
577
578   11)UserDao.java
579
580       package com.example.dao;
581
582       import java.util.List;
583       import com.example.vo.UserVO;
584
585       public interface UserDao {
586         void insert(UserVO user);
587
588         List<UserVO> readAll();
589
590         void update(UserVO user);
591
592         void delete(String id);
593
594         UserVO read(String id);
595       }
596
597   12)UserDaoImplJDBC.java
598       package com.example.dao;
599
600       import java.util.List;
601       import com.example.vo.UserVO;
602
603       public class UserDaoImplJDBC implements UserDao {
604
605         @Override
606         public void insert(UserVO user) {
607           // TODO Auto-generated method stub
608
609         }
610
611         @Override
612         public List<UserVO> readAll() {
613           // TODO Auto-generated method stub
614           return null;
615         }
616
617         @Override
618         public void update(UserVO user) {
```

```
619             // TODO Auto-generated method stub
620
621           }
622
623           @Override
624           public void delete(String id) {
625             // TODO Auto-generated method stub
626
627           }
628
629           @Override
630           public UserVO read(String id) {
631             // TODO Auto-generated method stub
632             return null;
633           }
634         }
635
636     13)Oracle Jdbc Driver 설치
637       <dependency>
638          <groupId>com.oracle</groupId>
639          <artifactId>ojdbc6</artifactId>
640          <version>11.1</version>
641         </dependency>
642
643       <참고>
644       -MySQL일 경우에는 'spring mysql'로 검색하여 MySQL Connector/J를 설치한다.
645         <dependency>
646             <groupId>mysql</groupId>
647             <artifactId>mysql-connector-java</artifactId>
648             <version>6.0.6</version>
649         </dependency>
650
651     14)Spring JDBC 설치
652       -JdbcTemplate를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.
653
654         <dependency>
655             <groupId>org.springframework</groupId>
656             <artifactId>spring-jdbc</artifactId>
657             <version>4.3.24.RELEASE</version>
658         </dependency>
659
660       -pom.xml에 붙여 넣고 Maven Install 하기
661
662     15)dbinfo.properties 파일 생성
663       -/src/main/resoures/dbinfo.properties 파일 생성
664       -/src/main/resources > right-click > New > File
665       -File name : dbinfo.properties > Finish
666
667         db.driverClass=oracle.jdbc.driver.OracleDriver
668         db.url=jdbc:oracle:thin:@localhost:1521:XE
669         db.username=scott
670         db.password=tiger
671
672       <참고>
```

673 -MySQL일 경우에는 다음과 같이 설정한다.
674 db.driverClass=com.mysql.jdbc.Driver
675 db.url=jdbc:mysql://192.168.136.5:3306/world
676 db.username=root
677 db.password=javamysql
678
679 16)Bean Configuration XML 작성
680 -/src/main/resources > right-click > New > Spring Bean Configuration File
681 -File name : beans.xml > Finish
682 -Namespace Tab
683 -Check context - http://www.springframework.org/schema/context
684
685 <?xml version="1.0" encoding="UTF-8"?>
686 <beans xmlns="http://www.springframework.org/schema/beans"
687 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
688 xmlns:context="http://www.springframework.org/schema/context"
689 xsi:schemaLocation="http://www.springframework.org/schema/beans
http://www.springframework.org/schema/beans/spring-beans.xsd
690 http://www.springframework.org/schema/context
http://www.springframework.org/schema/context/spring-context-3.2.xsd">
691
692 <context:property-placeholder location="classpath:dbinfo.properties" />
693 <bean id="dataSource"
class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
694 <property name="driverClass" value="${db.driverClass}" />
695 <property name="url" value="${db.url}" />
696 <property name="username" value="${db.username}" />
697 <property name="password" value="${db.password}" />
698 </bean>
699 </beans>
700
701 17)사용자 관리 프로젝트의 Bean 등록 및 의존 관계 설정
702 -<context:component-scan> 태그 사용
703 -@Service, @Repository 어노테이션을 선언한 클래스들과 @Autowired 어노테이션을 선언하여 의존관계를 설정한 클래스들이 위치한 패키지를 Scan하기 위한 설정을 XML에 해주어야 한다.
704 -beans.xml에 다음 코드 추가한다.
705
706 <context:component-scan base-package="com.example" />
707
708 18)Spring TestContext Framework 사용하기
709 -/src/test/java > right-click > New > JUnit Test Case
710 -Name : MembershipTest > Finish
711
712 import org.junit.Test;
713 import org.junit.runner.RunWith;
714 import org.springframework.beans.factory.annotation.Autowired;
715 import org.springframework.test.context.ContextConfiguration;
716 import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
717
718 import com.example.service.UserService;
719
720 @RunWith(SpringJUnit4ClassRunner.class)
721 @ContextConfiguration(locations="classpath:beans.xml")
722 public class MembershipTest {

```
723
724         @Autowired
725         UserService service;
726
727         @Test
728         public void test() {
729
730         }
731       }
732
733   19)Oracle JDBC Driver(ojdbc) Project BuildPath에 추가
734     -ojdbc6.jar  <--안해도 됨. 왜냐하면 이미 pom.xml에서 추가했기 때문
735
736     <참고>
737     -ojdbc8.jar(http://www.oracle.com/technetwork/database/features/jdbc/jdbc-ucp-122-3110062.html)
738     <참고>
739     -MySQL일 경우에는
        mysql-connector-java-5.1.42-bin.jar(https://dev.mysql.com/downloads/connector/j/) 추가
740
741
742 12. JDBC를 이용한 Membership Project
743   1)사용자 조회 테스트
744     -com.javasoft.dao.UserDaoImplJDBC.java 수정
745
746       @Repository("userDao")
747       public class UserDaoImplJDBC implements UserDao {
748         private DataSource dataSource;
749
750         @Autowired
751         public void setDataSource(DataSource dataSource) {
752           this.dataSource = dataSource;
753         }
754
755         ...
756         @Override
757         public UserVO read(String id) {
758           Connection conn = null;
759           PreparedStatement pstmt = null;
760           ResultSet rs = null;
761           UserVO userVO = null;
762           try {
763             conn = this.dataSource.getConnection();
764             pstmt = conn.prepareStatement("SELECT * FROM users WHERE userid = ?");
765             pstmt.setString(1, id);
766             rs = pstmt.executeQuery();
767             rs.next();
768             userVO = new UserVO(rs.getString("userid"), rs.getString("name"),
               rs.getString("gender"), rs.getString("city"));
769           }catch(SQLException ex) {
770             System.out.println(ex);
771           }finally {
772             try {
773               if(conn != null) conn.close();
```

```
774                 if(pstmt != null) pstmt.close();
775                 if(rs != null) rs.close();
776               }catch(SQLException ex) {
777                 System.out.println(ex);
778               }
779             }
780             return userVO;
781           }
782
783       -com.javasoft.service.UserServiceImpl.java 수정
784
785         @Service("userService")
786         public class UserServiceImpl implements UserService {
787
788           @Autowired
789           UserDao userDao;
790
791           ...
792           @Override
793           public UserVO getUser(String id) {
794             return userDao.read(id);
795           }
796
797       -/src/test/java/MembershipTest.java
798
799           @Test
800           public void test() {
801             //사용자 조회 테스트
802             UserVO user = service.getUser("jimin");
803             System.out.println(user);
804             assertEquals("한지민", user.getName());
805           }
806
807       ****************************************
808       java.lang.NoClassDefFoundError: Could not initialize class
          org.springframework.jdbc.core.StatementCreatorUtils
809       <에러 해결 방법>
810         <dependency>
811           <groupId>org.springframework</groupId>
812           <artifactId>spring-context</artifactId>
813           <version>4.3.24.RELEASE</version>
814         </dependency>
815
816       -버전 변경 후 Maven Clean  -> Maven Install
817
818     2)사용자 등록 및 목록 조회 테스트
819       -com.javasoft.dao.UserDaoImplJDBC.java 코드 수정
820         @Override
821         public void insert(UserVO user) {
822           Connection conn = null;
823           PreparedStatement pstmt = null;
824           try {
825             conn = this.dataSource.getConnection();
826             String sql = "INSERT INTO users (userid, name, gender,city) VALUES (?, ?, ?, ?)";
```

```
827             pstmt = conn.prepareStatement(sql);
828             pstmt.setString(1, user.getUserId());
829             pstmt.setString(2, user.getName());
830             pstmt.setString(3, user.getGender());
831             pstmt.setString(4, user.getCity());
832             pstmt.executeUpdate();
833             System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" +
                user.getName());
834           }catch(SQLException ex) {
835             System.out.println(ex);
836           }finally {
837             try {
838               if(conn != null) conn.close();
839               if(pstmt != null) pstmt.close();
840             }catch(SQLException ex) {
841               System.out.println(ex);
842             }
843           }
844         }
845
846         @Override
847         public List<UserVO> readAll() {
848           Connection conn = null;
849           Statement stmt = null;
850           ResultSet rs = null;
851           List<UserVO> userList = null;
852           try {
853             conn = this.dataSource.getConnection();
854             stmt = conn.createStatement();
855             rs = stmt.executeQuery("SELECT * FROM users");
856             userList = new ArrayList<UserVO>();
857             while(rs.next()) {
858               UserVO userVO = new UserVO(rs.getString("userid"), rs.getString("name"),
                  rs.getString("gender"), rs.getString("city"));
859               userList.add(userVO);
860             }
861           }catch(SQLException ex) {
862             System.out.println(ex);
863           }finally {
864             try {
865               if(conn != null) conn.close();
866               if(stmt != null) stmt.close();
867               if(rs != null) rs.close();
868             }catch(SQLException ex) {
869               System.out.println(ex);
870             }
871           }
872           return userList;
873         }
874
875       -com.javasoft.service.UserServiceImpl.java 코드 수정
876
877           @Override
878           public void insertUser(UserVO user) {
```

```
879             userDao.insert(user);
880           }
881
882           @Override
883           public List<UserVO> getUserList() {
884             return userDao.readAll();
885           }
886
887       -/src/test/java/MembershipTest.java
888
889           ...
890           @Test
891           public void test1() {
892             //사용자 등록 및 목록조회 테스트
893             this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
894             for(UserVO user : this.service.getUserList()){
895               System.out.println(user);
896             }
897           }
898
899       3)사용자 정보 수정 테스트
900         -com.javasoft.dao.UserDaoImplJDBC.java 코드 수정
901
902           @Override
903           public void update(UserVO user) {
904             Connection conn = null;
905             PreparedStatement pstmt = null;
906             try {
907               conn = this.dataSource.getConnection();
908               String sql = "UPDATE users  SET name = ?, gender = ?, city = ? WHERE userid = ?";
909               pstmt = conn.prepareStatement(sql);
910               pstmt.setString(1, user.getName());
911               pstmt.setString(2, user.getGender());
912               pstmt.setString(3, user.getCity());
913               pstmt.setString(4, user.getUserId());
914               pstmt.executeUpdate();
915               System.out.println("갱신된 Record with ID = " + user.getUserId() );
916             }catch(SQLException ex) {
917               System.out.println(ex);
918             }finally {
919               try {
920                 if(conn != null) conn.close();
921                 if(pstmt != null) pstmt.close();
922               }catch(SQLException ex) {
923                 System.out.println(ex);
924               }
925             }
926           }
927
928         -com.javasoft.service.UserServiceImpl.java 코드 수정
929
930           @Override
931           public void updateUser(UserVO user) {
932             userDao.update(user);
```

```
933         }
934
935     -/src/test/java/MembershipTest.java
936
937         @Ignore @Test
938         public void test1() {
939           //사용자 등록 및 목록조회 테스트
940           this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
941           for(UserVO user : this.service.getUserList()){
942             System.out.println(user);
943           }
944         }
945
946         @Test
947         public void test2() {
948           //사용자 정보 수정 테스트
949           service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
950           UserVO user = service.getUser("dooly");
951           System.out.println(user);
952         }
953
954     4)사용자 정보 삭제 테스트
955       -com.javasoft.dao.UserDaoImplJDBC.java 코드 수정
956
957         @Override
958         public void delete(String id) {
959           Connection conn = null;
960           PreparedStatement pstmt = null;
961           try {
962             conn = this.dataSource.getConnection();
963             pstmt = conn.prepareStatement("DELETE FROM users WHERE userid = ?");
964             pstmt.setString(1, id);
965             pstmt.executeUpdate();
966             System.out.println("삭제된 Record with ID = " + id );
967           }catch(SQLException ex) {
968             System.out.println(ex);
969           }finally {
970             try {
971               if(conn != null) conn.close();
972               if(pstmt != null) pstmt.close();
973             }catch(SQLException ex) {
974               System.out.println(ex);
975             }
976           }
977         }
978
979       -com.javasoft.service.UserServiceImpl.java 코드 수정
980
981         @Override
982         public void deleteUser(String id) {
983           userDao.delete(id);
984         }
985
986       -/src/test/java/MembershipTest.java
```

```
987         @Test
988         public void test() {
989           UserVO user = this.service.getUser("jimin");
990           System.out.println(user);
991           assertEquals("한지민", user.getName());
992         }
993         @Ignore @Test
994         public void test1() {
995           this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
996           for(UserVO user : this.service.getUserList()){
997             System.out.println(user);
998           }
999         }
1000
1001        @Ignore @Test
1002        public void test2() {
1003          service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1004          UserVO user = service.getUser("dooly");
1005          System.out.println(user);
1006        }
1007
1008        @Test
1009        public void test3() {
1010          //사용자 정보 삭제 테스트
1011          service.deleteUser("dooly");
1012          for(UserVO user : service.getUserList()){
1013            System.out.println(user);
1014          }
1015        }
1016
1017
1018  13. iBATIS를 이용한 Membership Project
1019    1)사용자 조회 테스트
1020      -Project Build path에 ibatis-2.3.4.726.jar 등록
1021
1022      -src/main/java/SqlMapConfig.xml 생성
1023        <?xml version="1.0" encoding="UTF-8"?>
1024        <!DOCTYPE sqlMapConfig
1025            PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"
1026              "http://ibatis.apache.org/dtd/sql-map-config-2.dtd">
1027        <sqlMapConfig>
1028          <properties resource="dbinfo.properties" />
1029          <settings useStatementNamespaces="true"/>
1030          <transactionManager type="JDBC">
1031            <dataSource type="SIMPLE">
1032              <property name="JDBC.Driver" value="${db.driverClass}"/>
1033              <property name="JDBC.ConnectionURL" value="${db.url}"/>
1034              <property name="JDBC.Username" value="${db.username}"/>
1035              <property name="JDBC.Password" value="${db.password}"/>
1036            </dataSource>
1037          </transactionManager>
1038          <sqlMap resource="com/javasoft/dao/Users.xml"/>
1039        </sqlMapConfig>
1040
```

-com.javasoft.dao/User.xml 생성

```
<?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"/>
  <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>
</sqlMap>
```

-com.javasoft.dao.UserDaoImplJDBC1.java 생성

```
@Repository("userDao1")
public class UserDaoImplJDBC1 implements UserDao {
  @Override
  public UserVO read(String id) {
    Reader rd = null;
    SqlMapClient smc = null;
    UserVO userVO = null;
    try {
      rd = Resources.getResourceAsReader("SqlMapConfig.xml");
      smc = SqlMapClientBuilder.buildSqlMapClient(rd);
      userVO = (UserVO)smc.queryForObject("Users.useResultMap", id);
    } catch (IOException | SQLException e) {
      // TODO Auto-generated catch block
      e.printStackTrace();
    }
    return userVO;
  }
}
```

-com.javasoft.service.UserServiceImpl.java 수정

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

    @Autowired
    UserDao userDao1;   //userDao에서 userDao1로 변경

    ...
    @Override
    public UserVO getUser(String id) {
      return userDao1.read(id);
    }
```

-/src/test/java/MembershipTest.java

```
1095          @Test
1096          public void test() {
1097            //사용자 조회 테스트
1098            UserVO user = service.getUser("jimin");
1099            System.out.println(user);
1100            assertEquals("한지민", user.getName());
1101          }
1102
1103      2)사용자 등록 및 목록 조회 테스트
1104        -Users.xml
1105          <insert id="insert" parameterClass="userVO">
1106            INSERT INTO USERS(userid, name, gender, city)
1107            VALUES (#userId#, #name#, #gender#, #city#)
1108          </insert>
1109
1110          <select id="getAll" resultClass="userVO">
1111            SELECT * FROM USERS
1112          </select>
1113
1114        -UserDaoImplJDBC1.java
1115          @Override
1116          public void insert(UserVO user) {
1117            Reader rd = null;
1118            SqlMapClient smc = null;
1119            UserVO userVO = null;
1120            try {
1121              rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1122              smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1123              smc.insert("Users.insert", user);
1124              System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" +
                  user.getName());
1125            } catch (IOException | SQLException e) {
1126              // TODO Auto-generated catch block
1127              e.printStackTrace();
1128            }
1129          }
1130
1131          @Override
1132          public List<UserVO> readAll() {
1133            Reader rd = null;
1134            SqlMapClient smc = null;
1135            List<UserVO> userList = null;
1136            try {
1137              rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1138              smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1139              userList = (List<UserVO>)smc.queryForList("Users.getAll", null);
1140            } catch (IOException | SQLException e) {
1141              // TODO Auto-generated catch block
1142              e.printStackTrace();
1143            }
1144            return userList;
1145          }
1146
1147        -MembershipTest.java
```

```
1148        @Test
1149        public void test1() {
1150          this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1151          for(UserVO user : this.service.getUserList()){
1152            System.out.println(user);
1153          }
1154        }
1155
1156    3)사용자 정보 수정 테스트
1157      -Users.xml
1158        <update id="update" parameterClass="userVO">
1159          UPDATE USERS
1160          SET    name = #name#, gender = #gender#, city = #city#
1161          WHERE  userId = #userId#
1162        </update>
1163
1164      -com.javasoft.dao.UserDaoImplJDBC1.java 코드 수정
1165        @Override
1166        public void update(UserVO user) {
1167          Reader rd = null;
1168          SqlMapClient smc = null;
1169          UserVO userVO = null;
1170          try {
1171            rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1172            smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1173            smc.update("Users.update", user);
1174            System.out.println("갱신된 Record with ID = " + user.getUserId() );
1175          } catch (IOException | SQLException e) {
1176            // TODO Auto-generated catch block
1177            e.printStackTrace();
1178          }
1179        }
1180
1181      -MembershipTest.java 수정
1182        @Ignore @Test
1183        public void test1() {
1184          this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1185          for(UserVO user : this.service.getUserList()){
1186            System.out.println(user);
1187          }
1188        }
1189
1190        @Test
1191        public void test2() {
1192          service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1193          UserVO user = service.getUser("dooly");
1194          System.out.println(user);
1195        }
1196
1197    4)사용자 정보 삭제 테스트
1198      -Users.xml
1199        <delete id="delete" parameterClass="String">
1200          DELETE FROM USERS WHERE  userid = #id#
1201        </delete>
```

```
1202
1203      -com.javasoft.dao.UserDaoImplJDBC.java 코드 수정
1204        @Override
1205        public void delete(String id) {
1206          Reader rd = null;
1207          SqlMapClient smc = null;
1208          UserVO userVO = null;
1209          try {
1210            rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1211            smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1212            smc.delete("Users.delete", id);
1213            System.out.println("삭제된 Record with ID = " + id );
1214          } catch (IOException | SQLException e) {
1215            // TODO Auto-generated catch block
1216            e.printStackTrace();
1217          }
1218        }
1219
1220      -MembershipTest.java 수정
1221        @Test
1222        public void test() {
1223          UserVO user = this.service.getUser("jimin");
1224          System.out.println(user);
1225          assertEquals("한지민", user.getName());
1226        }
1227        @Ignore @Test
1228        public void test1() {
1229          this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1230          for(UserVO user : this.service.getUserList()){
1231            System.out.println(user);
1232          }
1233        }
1234
1235        @Ignore @Test
1236        public void test2() {
1237          service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1238          UserVO user = service.getUser("dooly");
1239          System.out.println(user);
1240        }
1241
1242        @Test
1243        public void test3() {
1244          //사용자 정보 삭제 테스트
1245          service.deleteUser("dooly");
1246          for(UserVO user : service.getUserList()){
1247            System.out.println(user);
1248          }
1249        }
1250
1251
1252  14. MyBatis를 이용한 Membership Project
1253    1)사용자 조회 테스트
1254      -Project Build path에 mybatis-3.4.5.jar 등록
1255
```

```
-src/main/resources/dbinfo.properties
  db.driverClass=oracle.jdbc.driver.OracleDriver
  db.url=jdbc:oracle:thin:@localhost:1521:XE
  db.username=scott
  db.password=tiger

-src/main/java/mybatis-config.xml

  <?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/javasoft/dao/mybatis-mapper.xml"/>
    </mappers>
  </configuration>

-com.javasoft.dao/mybatis-mapper.xml

  <?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">
    <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>
  </mapper>

-UserServiceImpl.java 수정
  @Service("userService")
```

```
1309        public class UserServiceImpl implements UserService {
1310
1311          @Autowired
1312          UserDao userDao2;
1313
1314      -UserDaoImplJDBC2.java 수정
1315
1316          @Repository("userDao2")
1317          public class UserDaoImplJDBC2 implements UserDao {
1318            ...
1319            @Override
1320            public UserVO read(String id) {
1321              Reader rd = null;
1322              SqlSession session = null;
1323              UserVO userVO = null;
1324              try {
1325                rd = Resources.getResourceAsReader("mybatis-config.xml");
1326                session = session = new SqlSessionFactoryBuilder().build(rd).openSession();
1327                userVO = (UserVO)session.selectOne("select", id);
1328              } catch (IOException e) {
1329                // TODO Auto-generated catch block
1330                e.printStackTrace();
1331              }
1332              return userVO;
1333            }
1334
1335      -MembershipTest.java
1336
1337          @RunWith(SpringJUnit4ClassRunner.class)
1338          @ContextConfiguration(locations="classpath:beans.xml")
1339          public class MembershipTest {
1340
1341            @Autowired
1342            UserService service;
1343
1344            @Test
1345            public void test() {
1346              UserVO user = this.service.getUser("jimin");
1347              System.out.println(user);
1348              assertEquals("한지민", user.getName());
1349            }
1350
1351      2)사용자 등록 및 목록 조회 테스트
1352        -mybatis-mapper.xml
1353
1354          <insert id="insert" parameterType="userVO">
1355            INSERT INTO USERS(userid, name, gender, city)
1356            VALUES (#{userId}, #{name}, #{gender}, #{city})
1357          </insert>
1358
1359          <select id="selectAll" resultType="userVO" resultMap="userVOResult">
1360            SELECT * FROM USERS
1361          </select>
1362
```

-UserDaoImplJDBC2.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;
}
```

-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);
  }
}
```

3)사용자 정보 수정 테스트

-mybatis-mapper.xml

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

-UserDaoImplJDBC2.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();
  }
}
```

-MembershipTest.java

```
@Autowired
UserService service;

@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);
  }
}

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

4)사용자 정보 삭제 테스트

-mybatis-mapper.xml

```
1470        <delete id="delete" parameterType="String">
1471          DELETE FROM USERS WHERE  userid = #{id}
1472        </delete>
1473
1474      -UserDaoImplJDBC2.java
1475
1476        @Override
1477        public void delete(String id) {
1478          Reader rd = null;
1479          SqlSession session = null;
1480          UserVO userVO = null;
1481          try {
1482            rd = Resources.getResourceAsReader("mybatis-config.xml");
1483            session = new SqlSessionFactoryBuilder().build(rd).openSession();
1484            session.delete("delete", id);
1485            session.commit();
1486            System.out.println("삭제된 Record with ID = " + id );
1487          } catch (IOException e) {
1488            e.printStackTrace();
1489          }
1490        }
1491
1492      -MembershipTest.java
1493
1494        @Autowired
1495        UserService service;
1496
1497        @Test
1498        public void test() {
1499          UserVO user = this.service.getUser("jimin");
1500          System.out.println(user);
1501          assertEquals("한지민", user.getName());
1502        }
1503        @Ignore @Test
1504        public void test1() {
1505          this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1506          for(UserVO user : this.service.getUserList()){
1507            System.out.println(user);
1508          }
1509        }
1510
1511        @Ignore @Test
1512        public void test2() {
1513          service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1514          UserVO user = service.getUser("dooly");
1515          System.out.println(user);
1516        }
1517
1518        @Test
1519        public void test3() {
1520          //사용자 정보 삭제 테스트
1521          service.deleteUser("dooly");
1522          for(UserVO user : service.getUserList()){
1523            System.out.println(user);
```

```
1524          }
1525        }
1526
1527
1528  15. JDBCTemplate를 이용한 Membership Project
1529    1)사용자 조회 테스트
1530      -com.javasoft.dao.UserDaoImplJDBC.java 수정
1531
1532        @Repository("userDao")
1533        public class UserDaoImpleJDBC implements UserDao {
1534          private JdbcTemplate jdbcTemplate;
1535
1536          @Autowired
1537          public void setDataSource(DataSource dataSource) {
1538            this.jdbcTemplate = new JdbcTemplate(dataSource);
1539          }
1540
1541          class UserMapper implements RowMapper<UserVO> {
1542            public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
1543              UserVO user = new UserVO();
1544              user.setUserId(rs.getString("userid"));
1545              user.setName(rs.getString("name"));
1546              user.setGender(rs.getString("gender"));
1547              user.setCity(rs.getString("city"));
1548              return user;
1549            }
1550          }
1551
1552          ...
1553          @Override
1554          public UserVO read(String id) {
1555            String SQL = "SELECT * FROM users WHERE userid = ?";
1556            try {
1557              UserVO user = jdbcTemplate.queryForObject(SQL,
1558                   new Object[] { id }, new UserMapper());
1559              return user;
1560            }catch(EmptyResultDataAccessException e){
1561              return null;
1562            }
1563          }
1564
1565      -com.javasoft.service.UserServiceImpl.java 수정
1566
1567          @Service("userService")
1568          public class UserServiceImpl implements UserService {
1569
1570            @Autowired
1571            UserDao userdao;
1572
1573            ...
1574            @Override
1575            public UserVO getUser(String id) {
1576              return userdao.read(id);
1577            }
```

-/src/test/java/MembershipTest.java

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

****

java.lang.NoClassDefFoundError: Could not initialize class org.springframework.jdbc.core.StatementCreatorUtils

<에러 해결 방법>

```
<dependency>
<groupId>org.springframework</groupId>
<artifactId>spring-context</artifactId>
<version>4.3.13.RELEASE</version>
</dependency>
```

-버전 변경 후 Maven Clean -> Maven Install

2)사용자 등록 및 목록 조회 테스트

-com.javasoft.dao.UserDaoImplJDBC.java 코드 수정

```
@Override
public void insert(UserVO user) {
  String SQL = "INSERT INTO users (userid, name, gender,city) VALUES (?, ?, ?, ?)";
  jdbcTemplate.update(SQL, user.getUserId(), user.getName(), user.getGender(),
  user.getCity());

  System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" +
  user.getName());
}

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

-com.javasoft.service.UserServiceImpl.java 코드 수정

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

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

```
1629
1630      -/src/test/java/MembershipTest.java
1631
1632          @Test
1633          public void test1() {
1634            //사용자 등록 및 목록조회 테스트
1635            this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1636            for(UserVO user : this.service.getUserList()){
1637              System.out.println(user);
1638            }
1639          }
1640
1641      3)사용자 정보 수정 테스트
1642        -com.javasoft.dao.UserDaoImplJDBC.java 코드 수정
1643
1644          @Override
1645          public void update(UserVO user) {
1646            String SQL = "UPDATE users  SET name = ?, gender = ?, city = ? WHERE userid = ?";
1647            jdbcTemplate.update(SQL, user.getName(), user.getGender(),
                user.getCity(),user.getUserId());
1648            System.out.println("갱신된 Record with ID = " + user.getUserId() );
1649          }
1650
1651        -com.javasoft.service.UserServiceImpl.java 코드 수정
1652
1653          @Override
1654          public void updateUser(UserVO user) {
1655            userDao.update(user);
1656          }
1657
1658        -/src/test/java/MembershipTest.java
1659
1660          @Ignore @Test
1661          public void test1() {
1662            //사용자 등록 및 목록조회 테스트
1663            this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1664            for(UserVO user : this.service.getUserList()){
1665              System.out.println(user);
1666            }
1667          }
1668
1669          @Test
1670          public void test2() {
1671            //사용자 정보 수정 테스트
1672            service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1673            UserVO user = service.getUser("dooly");
1674            System.out.println(user);
1675          }
1676
1677      4)사용자 정보 삭제 테스트
1678        -com.javasoft.dao.UserDaoImplJDBC.java 코드 수정
1679
1680          @Override
1681          public void delete(String id) {
```

```
1682          String SQL = "DELETE FROM users WHERE userid = ?";
1683          jdbcTemplate.update(SQL, id);
1684          System.out.println("삭제된 Record with ID = " + id );
1685        }
1686
1687      -com.javasoft.service.UserServiceImpl.java 코드 수정
1688
1689        @Override
1690        public void deleteUser(String id) {
1691          userDao.delete(id);
1692        }
1693
1694      -/src/test/java/MembershipTest.java
1695
1696          @Test
1697          public void test3() {
1698            //사용자 정보 삭제 테스트
1699            service.deleteUser("dooly");
1700            for(UserVO user : service.getUserList()){
1701              System.out.println(user);
1702            }
1703          }
```