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 엔터프라이즈 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 ojdbc7'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 jdbc드라이버를 다운로드 받는다. -->ojdbc6.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.
mvn install:install-file
-Dfile="C:\oraclexe\app\oracle\product\11.2.0\server\jdbc\lib\ojdbc6.jar"
-DgroupId=com.oracle -DartifactId=ojdbc6 -Dversion=11.1 -Dpackaging=jar

인스톨 명령을 실행하면 메이븐 depository에 해당 드라이버가 설치된다.
위에서는 C:\Users\webnbiz01\.m2\repository\com\oracle\ojdbc6\11.1 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>ojdbc6</artifactId>
<version>11.1</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.javasoft Package 생성
-/src > right-click > New > Package
-Name : com.javasoft > Finish

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

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

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

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

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

    }
}
```

6)Maven Project로 전환

-SpringJdbcDemo Project > right-click > Configure > Convert to Maven Project
-Finish

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

8)Oracle Jdbc Driver 라이브러리 검색 및 설치

9)Spring Context 설치
-Maven Repository 에서 'Spring Context'로 검색하여 디펜던시 추가하고 설치

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

-pom.xml에 붙여 넣고 Maven Install 하기

10)Spring JDBC 설치
-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>
```

-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.javasoft.UserClient.java 코드 추가

```
package com.javasoft;

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.javasoft.vo

6)UserVO.java 생성

```
package com.javasoft.vo;
```

```
439
440     public class UserVO {
441
442         private String userId;
443         private String name;
444         private String gender;
445         private String city;
446
447         public UserVO() {}
448
449         public UserVO(String userId, String name, String gender, String city) {
450             this.userId = userId;
451             this.name = name;
452             this.gender = gender;
453             this.city = city;
454         }
455
456         public String getUserId() {
457             return userId;
458         }
459
460         public void setUserId(String userId) {
461             this.userId = userId;
462         }
463
464         public String getName() {
465             return name;
466         }
467
468         public void setName(String name) {
469             this.name = name;
470         }
471
472         public String getGender() {
473             return gender;
474         }
475
476         public void setGender(String gender) {
477             this.gender = gender;
478         }
479
480         public String getCity() {
481             return city;
482         }
483
484         public void setCity(String city) {
485             this.city = city;
486         }
487
488         @Override
489         public String toString() {
490             return "User [userId=" + userId + ", name=" + name + ", gender="
491                     + gender + ", city=" + city + "]";
492         }
493     }
494
495 7)/src/main/java > right-click > New > Package
496       Package name : com.javasoft.service
497
498 8)UserService.java
499
500     package com.javasoft.service;
501
502     import java.util.List;
503     import com.javasoft.vo.UserVO;
504
505     public interface UserService {
```

```
506
507         void insertUser(UserVO user);
508
509         List<UserVO> getUserList();
510
511         void deleteUser(String id);
512
513         UserVO getUser(String id);
514
515         void updateUser(UserVO user);
516       }
517
518     9)UserServiceImpl.java
519         package com.javasoft.service;
520
521         import java.util.List;
522         import com.javasoft.vo.UserVO;
523
524         public class UserServiceImpl implements UserService {
525
526             @Override
527             public void insertUser(UserVO user) {
528                 // TODO Auto-generated method stub
529
530             }
531
532             @Override
533             public List<UserVO> getUserList() {
534                 // TODO Auto-generated method stub
535                 return null;
536             }
537
538             @Override
539             public void deleteUser(String id) {
540                 // TODO Auto-generated method stub
541
542             }
543
544             @Override
545             public UserVO getUser(String id) {
546                 // TODO Auto-generated method stub
547                 return null;
548             }
549
550             @Override
551             public void updateUser(UserVO user) {
552                 // TODO Auto-generated method stub
553
554             }
555         }
556
557     10)/src/main/java > right-click > New > Package
558         Package name : com.javasoft.dao
559
560     11)UserDao.java
561
562         package com.javasoft.dao;
563
564         import java.util.List;
565         import com.javasoft.vo.UserVO;
566
567         public interface UserDao {
568             void insert(UserVO user);
569
570             List<UserVO> readAll();
571
572             void update(UserVO user);
```

```
573
574             void delete(String id);
575
576             UserVO read(String id);
577          }
578
579       12)UserDaoImplJDBC.java
580          package com.javasoft.dao;
581
582          import java.util.List;
583          import com.javasoft.vo.UserVO;
584
585          public class UserDaoImplJDBC implements UserDao {
586
587             @Override
588             public void insert(UserVO user) {
589                // TODO Auto-generated method stub
590
591             }
592
593             @Override
594             public List<UserVO> readAll() {
595                // TODO Auto-generated method stub
596                return null;
597             }
598
599             @Override
600             public void update(UserVO user) {
601                // TODO Auto-generated method stub
602
603             }
604
605             @Override
606             public void delete(String id) {
607                // TODO Auto-generated method stub
608
609             }
610
611             @Override
612             public UserVO read(String id) {
613                // TODO Auto-generated method stub
614                return null;
615             }
616          }
617
618       13)Oracle Jdbc Driver 설치
619          <dependency>
620             <groupId>com.oracle</groupId>
621             <artifactId>ojdbc6</artifactId>
622             <version>11.1</version>
623             </dependency>
624
625          <참고>
626          -MySQL일 경우에는 'spring mysql'로 검색하여 MySQL Connector/J를 설치한다.
627             <dependency>
628                   <groupId>mysql</groupId>
629                   <artifactId>mysql-connector-java</artifactId>
630                   <version>6.0.6</version>
631             </dependency>
632
633       14)Spring JDBC 설치
634          -JdbcTemplate를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.
635
636             <dependency>
637                   <groupId>org.springframework</groupId>
638                   <artifactId>spring-jdbc</artifactId>
639                   <version>4.3.13.RELEASE</version>
```

```
        </dependency>

    -pom.xml에 붙여 넣고 Maven Install 하기

15)dbinfo.properties 파일 생성
    -/src/main/resoures/dbinfo.properties 파일 생성
    -/src/main/resources > right-click > New > File
    -File name : dbinfo.properties > Finish

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

    <참고>
    -MySQL일 경우에는 다음과 같이 설정한다.
        db.driverClass=com.mysql.jdbc.Driver
        db.url=jdbc:mysql://192.168.136.5:3306/world
        db.username=root
        db.password=javamysql

16)Bean Configuration XML 작성
    -/src/main/resources > right-click > New > Spring Bean Configuration File
    -File name : beans.xml > Finish
    -Namespace Tab
    -Check context - http://www.springframework.org/schema/context

        <?xml version="1.0" encoding="UTF-8"?>
        <beans xmlns="http://www.springframework.org/schema/beans"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xmlns:context="http://www.springframework.org/schema/context"
            xsi:schemaLocation="http://www.springframework.org/schema/beans
            http://www.springframework.org/schema/beans/spring-beans.xsd
                http://www.springframework.org/schema/context
                http://www.springframework.org/schema/context/spring-context-3.2.xsd">

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

17)사용자 관리 프로젝트의 Bean 등록 및 의존 관계 설정
    -<context:component-scan> 태그 사용
    -@Service, @Repository 어노테이션을 선언한 클래스들과 @Autowired 어노테이션을 선언하여
    의존관계를 설정한 클래스들이 위치한 패키지를 Scan하기 위한 설정을 XML에 해주어야 한다.
    -beans.xml에 다음 코드 추가한다.

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

18)Spring TestContext Framework 사용하기
    -/src/test/java > right-click > New > JUnit Test Case
    -Name : MembershipTest > Finish

        import org.junit.Test;
        import org.junit.runner.RunWith;
        import org.springframework.beans.factory.annotation.Autowired;
        import org.springframework.test.context.ContextConfiguration;
        import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;

        import com.javasoft.service.UserService;

        @RunWith(SpringJUnit4ClassRunner.class)
```

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

   @Autowired
   UserService service;

   @Test
   public void test() {

   }
}
```

19)Oracle JDBC Driver(ojdbc) Project BuildPath에 추가
-ojdbc6.jar

<참고>

-ojdbc8.jar(http://www.oracle.com/technetwork/database/features/jdbc/jdbc-ucp-122-3110062.html)
<참고>
-MySQL일 경우에는
mysql-connector-java-5.1.42-bin.jar(https://dev.mysql.com/downloads/connector/j/) 추가

12. JDBC를 이용한 Membership Project
1)사용자 조회 테스트
-com.javasoft.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.javasoft.service.UserServiceImpl.java 수정

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

   @Autowired
   UserDao userDao;

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

-/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) {
      Connection conn = null;
      PreparedStatement pstmt = null;
      try {
         conn = this.dataSource.getConnection();
         String sql = "INSERT INTO users (userid, name, gender,city) VALUES (?, ?, ?, ?)";
         pstmt = conn.prepareStatement(sql);
         pstmt.setString(1, user.getUserId());
         pstmt.setString(2, user.getName());
         pstmt.setString(3, user.getGender());
         pstmt.setString(4, user.getCity());
         pstmt.executeUpdate();
         System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" +
         user.getName());
      }catch(SQLException ex) {
         System.out.println(ex);
      }finally {
         try {
            if(conn != null) conn.close();
            if(pstmt != null) pstmt.close();
         }catch(SQLException ex) {
            System.out.println(ex);
         }
      }

      @Override
      public List<UserVO> readAll() {
         Connection conn = null;
         Statement stmt = null;
```

```
            ResultSet rs = null;
            List<UserVO> userList = null;
            try {
               conn = this.dataSource.getConnection();
               stmt = conn.createStatement();
               rs = stmt.executeQuery("SELECT * FROM users");
               userList = new ArrayList<UserVO>();
               while(rs.next()) {
                  UserVO userVO = new UserVO(rs.getString("userid"), rs.getString("name"),
                  rs.getString("gender"), rs.getString("city"));
                  userList.add(userVO);
               }
            }catch(SQLException ex) {
               System.out.println(ex);
            }finally {
               try {
                  if(conn != null) conn.close();
                  if(stmt != null) stmt.close();
                  if(rs != null) rs.close();
               }catch(SQLException ex) {
                  System.out.println(ex);
               }
            }
            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();
            }
```

-/src/test/java/MembershipTest.java

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

3)사용자 정보 수정 테스트

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

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

```
            }catch(SQLException ex) {
               System.out.println(ex);
            }finally {
               try {
                  if(conn != null) conn.close();
                  if(pstmt != null) pstmt.close();
               }catch(SQLException ex) {
                  System.out.println(ex);
               }
            }
         }
```

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

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

-/src/test/java/MembershipTest.java

```
            @Ignore @Test
            public void test1() {
               //사용자 등록 및 목록조회 테스트
               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 테스트
               service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
               UserVO user = service.getUser("dooly");
               System.out.println(user);
            }
```

4)사용자 정보 삭제 테스트

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

```
         @Override
         public void delete(String id) {
            Connection conn = null;
            PreparedStatement pstmt = null;
            try {
               conn = this.dataSource.getConnection();
               pstmt = conn.prepareStatement("DELETE FROM users WHERE userid = ?");
               pstmt.setString(1, id);
               pstmt.executeUpdate();
               System.out.println("삭제된 Record with ID = " + id );
            }catch(SQLException ex) {
               System.out.println(ex);
            }finally {
               try {
                  if(conn != null) conn.close();
                  if(pstmt != null) pstmt.close();
               }catch(SQLException ex) {
                  System.out.println(ex);
               }
            }
         }
```

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

```
         @Override
         public void deleteUser(String id) {
```

```
            userDao.delete(id);
        }

    -/src/test/java/MembershipTest.java
        @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 테스트
            service.deleteUser("dooly");
            for(UserVO user : service.getUserList()){
                System.out.println(user);
            }
        }

13. iBATIS를 이용한 Membership Project
  1)사용자 조회 테스트
    -Project Build path에 ibatis-2.3.4.726.jar 등록

    -src/main/java/SqlMapConfig.xml 생성
      <?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/javasoft/dao/Users.xml"/>
      </sqlMapConfig>

    -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.javasoft.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

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


2)사용자 등록 및 목록 조회 테스트
    -Users.xml
        <insert id="insert" parameterClass="userVO">
            INSERT INTO USERS(userid, name, gender, city)
            VALUES (#userId#, #name#, #gender#, #city#)
        </insert>

        <select id="getAll" resultClass="userVO">
            SELECT * FROM USERS
        </select>

    -UserDaoImplJDBC1.java
        @Override
        public void insert(UserVO user) {
```

```
            Reader rd = null;
            SqlMapClient smc = null;
            UserVO userVO = null;
            try {
               rd = Resources.getResourceAsReader("SqlMapConfig.xml");
               smc = SqlMapClientBuilder.buildSqlMapClient(rd);
               smc.insert("Users.insert", user);
               System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" +
               user.getName());
            } catch (IOException | SQLException e) {
               // TODO Auto-generated catch block
               e.printStackTrace();
            }
         }

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

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

   3)사용자 정보 수정 테스트
      -Users.xml
         <update id="update" parameterClass="userVO">
            UPDATE USERS
            SET    name = #name#, gender = #gender#, city = #city#
            WHERE  userId = #userId#
         </update>

      -com.javasoft.dao.UserDaoImplJDBC1.java 코드 수정
         @Override
         public void update(UserVO user) {
            Reader rd = null;
            SqlMapClient smc = null;
            UserVO userVO = null;
            try {
               rd = Resources.getResourceAsReader("SqlMapConfig.xml");
               smc = SqlMapClientBuilder.buildSqlMapClient(rd);
               smc.update("Users.update", user);
               System.out.println("갱신된 Record with ID = " + user.getUserId() );
            } catch (IOException | SQLException e) {
               // TODO Auto-generated catch block
               e.printStackTrace();
            }
         }

      -MembershipTest.java 수정
         @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

-Users.xml

```
    <delete id="delete" parameterClass="String">
       DELETE FROM USERS WHERE  userid = #id#
    </delete>
```

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

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

-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 테스트
       service.deleteUser("dooly");
       for(UserVO user : service.getUserList()){
          System.out.println(user);
       }
    }
```

14. MyBatis를 이용한 Membership Project

1)사용자 조회 테스트

-Project Build path에 mybatis-3.4.5.jar 등록

-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.javasoft.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.javasoft.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")
public class UserServiceImpl implements UserService {

  @Autowired
  UserDao userDao2;
```

-UserDaoImplJDBC2.java 수정

```
@Repository("userDao2")
```

```
public class UserDaoImplJDBC2 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 = session = new SqlSessionFactoryBuilder().build(rd).openSession();
            userVO = (UserVO)session.selectOne("select", id);
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
        return userVO;
    }
```

-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());
    }
```

2)사용자 등록 및 목록 조회 테스트

-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>
```

-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

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

-UserDaoImplJDBC2.java

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

-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);
        }
    }

    @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 테스트
            service.deleteUser("dooly");
            for(UserVO user : service.getUserList()){
                System.out.println(user);
            }
        }
```

15. JDBCTemplate를 이용한 Membership Project

1)사용자 조회 테스트

-com.javasoft.dao.UserDaoImplJDBC.java 수정

```
        @Repository("userDao")
        public class UserDaoImpleJDBC implements UserDao {
            private JdbcTemplate jdbcTemplate;

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

            class UserMapper implements RowMapper<UserVO> {
                public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
                    UserVO user = new UserVO();
                    user.setUserId(rs.getString("userid"));
                    user.setName(rs.getString("name"));
                    user.setGender(rs.getString("gender"));
                    user.setCity(rs.getString("city"));
                    return user;
                }
            }

            ...
            @Override
            public UserVO read(String id) {
                String SQL = "SELECT * FROM users WHERE userid = ?";
                try {
                    UserVO user = jdbcTemplate.queryForObject(SQL,
                            new Object[] { id }, new UserMapper());
                    return user;
                }catch(EmptyResultDataAccessException e){
                    return null;
                }
            }
```

-com.javasoft.service.UserServiceImpl.java 수정

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

            @Autowired
            UserDao userdao;

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

-/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();
        }

    -/src/test/java/MembershipTest.java

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

3)사용자 정보 수정 테스트
    -com.javasoft.dao.UserDaoImplJDBC.java 코드 수정

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

```
            System.out.println("갱신된 Record with ID = " + user.getUserId() );
        }

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

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

-/src/test/java/MembershipTest.java

        @Ignore @Test
        public void test1() {
            //사용자 등록 및 목록조회 테스트
            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 테스트
            service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
            UserVO user = service.getUser("dooly");
            System.out.println(user);
        }

4)사용자 정보 삭제 테스트
-com.javasoft.dao.UserDaoImplJDBC.java 코드 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 );
    }

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

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

-/src/test/java/MembershipTest.java

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