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>4.3.24.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
```

-Name : com.example > 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=hr
db.password=hr
```

5)/src/com.example.UserClient.java 생성
-/src > com.example > 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)pom.xml에 Oracle Jdbc Driver 설정하기

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

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

-pom.xml에 붙여 넣고 Maven Install 하기

10)Spring JDBC 설치
-JdbcTemplate를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.

```
350
351         <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
352         <dependency>
353             <groupId>org.springframework</groupId>
354             <artifactId>spring-jdbc</artifactId>
355             <version>4.3.24.RELEASE</version>
356         </dependency>
357
358       -pom.xml에 붙여 넣고 Maven Install 하기
359         [INFO] BUILD SUCCESS
360
361     11)Bean Configuration XML 작성
362       -/src/config > right-click > New > Other > Spring > Spring Bean Configuration File
363       -File name : beans.xml > Next
364       -Check [beans - http://www.springframework.org/schema/beans]
365       -Check [http://www.springframework.org/schema/beans/spring-beans-4.3.xsd]
366       -Finish
367
368         <?xml version="1.0" encoding="UTF-8"?>
369         <beans xmlns="http://www.springframework.org/schema/beans"
370         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
371         xsi:schemaLocation="http://www.springframework.org/schema/beans
372         http://www.springframework.org/schema/beans/spring-beans.xsd">
373
374         </beans>
375
376       -Namespace tab에서 context - http://www.springframework.org/schema/context check
377
378         <context:property-placeholder location="classpath:dbinfo.properties" />
379         <bean id="dataSource"
          class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
380           <property name="driverClass" value="${db.driverClass}" />
381           <property name="url" value="${db.url}" />
382           <property name="username" value="${db.username}" />
383           <property name="password" value="${db.password}" />
384         </bean>
385
386     12)/src/com.example.UserClient.java 코드 추가
387
388         package com.example;
389
390         import java.sql.SQLException;
391
392         import javax.sql.DataSource;
393
394         import org.springframework.context.ApplicationContext;
395         import org.springframework.context.support.GenericXmlApplicationContext;
396
397         public class UserClient {
398           public static void main(String[] args) {
399             ApplicationContext ctx = new GenericXmlApplicationContext("classpath:beans.xml");
400
401             DataSource ds = (DataSource) ctx.getBean("dataSource");
402             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.png 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)In Package Explorer > right-click > New > Java Project
    -Project name : Membership

```

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

6)com.example.vo.UserVO.java 생성

```
package com.example.vo;

public class UserVO {

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

  public UserVO() {}

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

  public String getUserId() {
    return userId;
  }

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

  public String getName() {
    return name;
  }

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

  public String getGender() {
    return gender;
  }

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

  public String getCity() {
    return city;
  }

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

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

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

```
619         }
620
621         @Override
622         public List<UserVO> readAll() {
623           // TODO Auto-generated method stub
624           return null;
625         }
626
627         @Override
628         public void update(UserVO user) {
629           // TODO Auto-generated method stub
630
631         }
632
633         @Override
634         public void delete(String id) {
635           // TODO Auto-generated method stub
636
637         }
638
639         @Override
640         public UserVO read(String id) {
641           // TODO Auto-generated method stub
642           return null;
643         }
644       }
645
646   13)Java Project를 Spring Project로 변환
647     -Membership Project > right-click > Configuration > Convert to Maven Project
648       --Project : /Membership
649       --Group Id : Membership
650       --Artifact Id : Membership
651       --version : 0.0.1-SNAPSHOT
652       --Packaging : jar
653       --Finish
654       --Package Explorer에서 보이는 Project 아이콘에 Maven의 'M'자가 보임.
655
656       -Membership Project > right-click > Spring > Add Spring Project Nature
657         --Package Explorer에서 보이는 Project 아이콘에 'M'자와 Spring의 'S'가 보임.
658
659       -pom.xml 파일에 Spring Context Dependency 추가하기
660         <version>0.0.1-SNAPSHOT</version>
661         <dependencies> <--- dependencies element 추가
662           <dependency> <---여기에 paste
663             <groupId>org.springframework</groupId>
664             <artifactId>spring-context</artifactId>
665             <version>4.3.24.RELEASE</version>
666           </dependency>
667         </dependencies>
668
669       -pom.xml > right-click > Run As > Maven install
670         [INFO] BUILD SUCCESS 확인
671
672   14)Oracle Jdbc Driver 설치
```

```
673     <dependency>
674        <groupId>com.oracle</groupId>
675        <artifactId>ojdbc8</artifactId>
676        <version>12.2</version>
677       </dependency>
678
679     <참고>
680     -MySQL일 경우에는 'spring mysql'로 검색하여 MySQL Connector/J를 설치한다.
681       <dependency>
682           <groupId>mysql</groupId>
683           <artifactId>mysql-connector-java</artifactId>
684           <version>6.0.6</version>
685       </dependency>
686
687   15)Spring JDBC 설치
688     -JdbcTemplate를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.
689
690       <dependency>
691           <groupId>org.springframework</groupId>
692           <artifactId>spring-jdbc</artifactId>
693           <version>4.3.24.RELEASE</version>
694       </dependency>
695
696     -pom.xml에 붙여 넣고 Maven Install 하기
697       [INFO] BUILD SUCCESS 확인
698
699   16)resource folder 생성
700     -Membership project > right-click > Build Path > Coinfigure Build Path
701     -Source Tab > Add Folder > Select Membership project > Click [Create New Folder] button
702     -Folder name : resources > Finish > OK
703     -Apply and Close
704
705   17)dbinfo.properties 파일 생성
706     -/resources > right-click > New > File
707     -File name : dbinfo.properties > Finish
708
709       db.driverClass=oracle.jdbc.driver.OracleDriver
710       db.url=jdbc:oracle:thin:@localhost:1521:XE
711       db.username=hr
712       db.password=hr
713
714     <참고>
715     -MySQL일 경우에는 다음과 같이 설정한다.
716       db.driverClass=com.mysql.jdbc.Driver
717       db.url=jdbc:mysql://192.168.136.5:3306/world
718       db.username=root
719       db.password=javamysql
720
721   18)Bean Configuration XML 작성
722     -/resources > right-click > New > Spring Bean Configuration File
723     -File name : beans.xml > Finish
724     -Namespace Tab
725     -Check context - http://www.springframework.org/schema/context
726
```

```
727         <?xml version="1.0" encoding="UTF-8"?>
728         <beans xmlns="http://www.springframework.org/schema/beans"
729           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
730           xmlns:context="http://www.springframework.org/schema/context"
731           xsi:schemaLocation="http://www.springframework.org/schema/beans
              http://www.springframework.org/schema/beans/spring-beans.xsd
732             http://www.springframework.org/schema/context
                http://www.springframework.org/schema/context/spring-context-3.2.xsd">
733
734           <context:property-placeholder location="classpath:dbinfo.properties" />
735           <bean id="dataSource"
              class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
736             <property name="driverClass" value="${db.driverClass}" />
737             <property name="url" value="${db.url}" />
738             <property name="username" value="${db.username}" />
739             <property name="password" value="${db.password}" />
740           </bean>
741         </beans>
742
743     19)사용자 관리 프로젝트의 Bean 등록 및 의존 관계 설정
744       -<context:component-scan> 태그 사용
745       -@Service, @Repository 어노테이션을 선언한 클래스들과 @Autowired 어노테이션을 선언하여 의존관계를 설정
          한 클래스들이 위치한 패키지를 Scan하기 위한 설정을 XML에 해주어야 한다.
746       -beans.xml에 다음 코드 추가한다.
747
748         <context:component-scan base-package="com.example" />
749
750     20)Spring TestContext Framework 사용하기
751       -/src > right-click > New > Package
752       -Package Name : com.example.test
753       -com.example.test > right-click > New > JUnit Test Case
754       -Name : MembershipTest > Finish
755       -Select [Not now] > OK
756
757         package com.example.test;
758
759         import org.junit.Test;
760         import org.junit.runner.RunWith;
761         import org.springframework.beans.factory.annotation.Autowired;
762         import org.springframework.test.context.ContextConfiguration;
763         import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
764
765         import com.example.service.UserService;
766
767         @RunWith(SpringJUnit4ClassRunner.class)
768         @ContextConfiguration(locations="classpath:beans.xml")
769         public class MembershipTest {
770
771           @Autowired
772           UserService service;
773
774           @Test
775           public void test() {
776
```

```
        }
      }

  21)Oracle JDBC Driver(ojdbc) Project BuildPath에 추가
    -ojdbc6.jar  <--안해도 됨. 왜냐하면 이미 pom.xml에서 추가했기 때문

    <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.example.dao.UserDaoImplJDBC.java 수정

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

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

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

```
828         }
829
830     -com.example.service.UserServiceImpl.java 수정
831
832       @Service("userService")
833       public class UserServiceImpl implements UserService {
834
835         @Autowired
836         UserDao userDao;
837
838         ...
839         @Override
840         public UserVO getUser(String id) {
841           return userDao.read(id);
842         }
843
844     -com.example.test.MembershipTest.java
845
846         @Test
847         public void test() {
848           //사용자 조회 테스트
849           UserVO user = service.getUser("jimin");
850           System.out.println(user);
851           assertEquals("한지민", user.getName());
852         }
853
854     -right-click > Run As > Junit Test
855     -결과 -> Junit View에 초록색 bar
856       UserVO [userId=jimin, name=한지민, gender=여, city=서울]
857
858   2)사용자 등록 및 목록 조회 테스트
859     -com.example.dao.UserDaoImplJDBC.java 코드 수정
860       @Override
861       public void insert(UserVO user) {
862         Connection conn = null;
863         PreparedStatement pstmt = null;
864         try {
865           conn = this.dataSource.getConnection();
866           String sql = "INSERT INTO users (userid, name, gender,city) VALUES (?, ?, ?, ?)";
867           pstmt = conn.prepareStatement(sql);
868           pstmt.setString(1, user.getUserId());
869           pstmt.setString(2, user.getName());
870           pstmt.setString(3, user.getGender());
871           pstmt.setString(4, user.getCity());
872           pstmt.executeUpdate();
873           System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" +
              user.getName());
874         }catch(SQLException ex) {
875           System.out.println(ex);
876         }finally {
877           try {
878             if(conn != null) conn.close();
879             if(pstmt != null) pstmt.close();
880           }catch(SQLException ex) {
```

```
881             System.out.println(ex);
882           }
883         }
884       }
885
886       @Override
887       public List<UserVO> readAll() {
888         Connection conn = null;
889         Statement stmt = null;
890         ResultSet rs = null;
891         List<UserVO> userList = null;
892         try {
893           conn = this.dataSource.getConnection();
894           stmt = conn.createStatement();
895           rs = stmt.executeQuery("SELECT * FROM users");
896           userList = new ArrayList<UserVO>();
897           while(rs.next()) {
898             UserVO userVO = new UserVO(rs.getString("userid"), rs.getString("name"),
                rs.getString("gender"), rs.getString("city"));
899             userList.add(userVO);
900           }
901         }catch(SQLException ex) {
902           System.out.println(ex);
903         }finally {
904           try {
905             if(conn != null) conn.close();
906             if(stmt != null) stmt.close();
907             if(rs != null) rs.close();
908           }catch(SQLException ex) {
909             System.out.println(ex);
910           }
911         }
912         return userList;
913       }
914
915     -com.example.service.UserServiceImpl.java 코드 수정
916
917         @Override
918         public void insertUser(UserVO user) {
919           userDao.insert(user);
920         }
921
922         @Override
923         public List<UserVO> getUserList() {
924           return userDao.readAll();
925         }
926
927     -com.example.test.MembershipTest.java
928
929         ...
930         @Test
931         public void test1() {
932           //사용자 등록 및 목록조회 테스트
933           this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
```

```
934             for(UserVO user : this.service.getUserList()){
935               System.out.println(user);
936             }
937           }
938
939       -right-click > Run As > Junit Test
940       -결과 -> Junit View에 초록색 bar
941       UserVO [userId=jimin, name=한지민, gender=여, city=서울]
942       등록된 Record UserId=dooly Name=둘리
943       UserVO [userId=dooly, name=둘리, gender=남, city=경기]
944       UserVO [userId=jimin, name=한지민, gender=여, city=서울]
945
946
947     3)사용자 정보 수정 테스트
948       -com.example.dao.UserDaoImplJDBC.java 코드 수정
949
950         @Override
951         public void update(UserVO user) {
952           Connection conn = null;
953           PreparedStatement pstmt = null;
954           try {
955             conn = this.dataSource.getConnection();
956             String sql = "UPDATE users  SET name = ?, gender = ?, city = ? WHERE userid = ?";
957             pstmt = conn.prepareStatement(sql);
958             pstmt.setString(1, user.getName());
959             pstmt.setString(2, user.getGender());
960             pstmt.setString(3, user.getCity());
961             pstmt.setString(4, user.getUserId());
962             pstmt.executeUpdate();
963             System.out.println("갱신된 Record with ID = " + user.getUserId() );
964           }catch(SQLException ex) {
965             System.out.println(ex);
966           }finally {
967             try {
968               if(conn != null) conn.close();
969               if(pstmt != null) pstmt.close();
970             }catch(SQLException ex) {
971               System.out.println(ex);
972             }
973           }
974         }
975
976       -com.example.service.UserServiceImpl.java 코드 수정
977
978           @Override
979           public void updateUser(UserVO user) {
980             userDao.update(user);
981           }
982
983       -com.example.test.MembershipTest.java
984
985           @Ignore @Test
986           public void test1() {
987             //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);
          }

    -right-click > Run As > Junit Test
    -결과 -> Junit View에 초록색 bar
    UserVO [userId=jimin, name=한지민, gender=여, city=서울]
    갱신된 Record with ID = dooly
    UserVO [userId=dooly, name=김둘리, gender=여, city=부산]

  4)사용자 정보 삭제 테스트
    -com.example.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.example.service.UserServiceImpl.java 코드 수정

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

    -com.example.test.MembershipTest.java
        @Test
```

```
1042          public void test() {
1043            UserVO user = this.service.getUser("jimin");
1044            System.out.println(user);
1045            assertEquals("한지민", user.getName());
1046          }
1047          @Ignore @Test
1048          public void test1() {
1049            this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1050            for(UserVO user : this.service.getUserList()){
1051              System.out.println(user);
1052            }
1053          }
1054
1055          @Ignore @Test
1056          public void test2() {
1057            service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1058            UserVO user = service.getUser("dooly");
1059            System.out.println(user);
1060          }
1061
1062          @Test
1063          public void test3() {
1064            //사용자 정보 삭제 테스트
1065            service.deleteUser("dooly");
1066            for(UserVO user : service.getUserList()){
1067              System.out.println(user);
1068            }
1069          }
1070
1071      -right-click > Run As > Junit Test
1072      -결과 -> Junit View에 초록색 bar
1073      UserVO [userId=jimin, name=한지민, gender=여, city=서울]
1074      삭제된 Record with ID = dooly
1075      UserVO [userId=jimin, name=한지민, gender=여, city=서울]
1076
1077
1078  13. iBATIS를 이용한 Membership Project
1079    1)준비
1080      -mvnrepository(https://mvnrepository.com에서 'ibatis'로 검색
1081      -Ibatis Sqlmap에서 2.3.4.726으로 들어가서 아래의 코드를 복사해서 pom.xml에 넣기
1082        <dependency>
1083          <groupId>org.apache.ibatis</groupId>
1084          <artifactId>ibatis-sqlmap</artifactId>
1085          <version>2.3.4.726</version>
1086        </dependency>
1087
1088      -pom.xml에 붙여 넣고 Maven Install 하기
1089        [INFO] BUILD SUCCESS 확인
1090
1091      -SqlMapConfig.xml 생성
1092        --src > right-click > New > Other > XML > XML File > Next
1093        --File name : SqlMapConfig.xml > Finish
1094        --<!DOCTYPE element는 internet에서 sqlmapconfig.xml로 검색
1095
```

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE sqlMapConfig
    PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"
      "http://ibatis.apache.org/dtd/sql-map-config-2.dtd">
<sqlMapConfig>
  <properties resource="dbinfo.properties" />
  <settings useStatementNamespaces="true"/>
  <transactionManager type="JDBC">
    <dataSource type="SIMPLE">
      <property name="JDBC.Driver" value="${db.driverClass}"/>
      <property name="JDBC.ConnectionURL" value="${db.url}"/>
      <property name="JDBC.Username" value="${db.username}"/>
      <property name="JDBC.Password" value="${db.password}"/>
    </dataSource>
  </transactionManager>
  <sqlMap resource="com/example/dao/Users.xml"/>
</sqlMapConfig>
```

2)사용자 조회 테스트

-com.example.dao/Users.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.example.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.example.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());
      }

3)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();
      }
```

```
1203        }
1204
1205        @Override
1206        public List<UserVO> readAll() {
1207          Reader rd = null;
1208          SqlMapClient smc = null;
1209          List<UserVO> userList = null;
1210          try {
1211            rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1212            smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1213            userList = (List<UserVO>)smc.queryForList("Users.getAll", null);
1214          } catch (IOException | SQLException e) {
1215            // TODO Auto-generated catch block
1216            e.printStackTrace();
1217          }
1218          return userList;
1219        }
1220
1221      -MembershipTest.java
1222        @Test
1223        public void test1() {
1224          this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1225          for(UserVO user : this.service.getUserList()){
1226            System.out.println(user);
1227          }
1228        }
1229
1230    4)사용자 정보 수정 테스트
1231      -Users.xml
1232        <update id="update" parameterClass="userVO">
1233          UPDATE USERS
1234          SET    name = #name#, gender = #gender#, city = #city#
1235          WHERE  userId = #userId#
1236        </update>
1237
1238      -com.example.dao.UserDaoImplJDBC1.java 코드 수정
1239        @Override
1240        public void update(UserVO user) {
1241          Reader rd = null;
1242          SqlMapClient smc = null;
1243          UserVO userVO = null;
1244          try {
1245            rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1246            smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1247            smc.update("Users.update", user);
1248            System.out.println("갱신된 Record with ID = " + user.getUserId() );
1249          } catch (IOException | SQLException e) {
1250            // TODO Auto-generated catch block
1251            e.printStackTrace();
1252          }
1253        }
1254
1255      -MembershipTest.java 수정
1256        @Ignore @Test
```

```
1257        public void test1() {
1258          this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1259          for(UserVO user : this.service.getUserList()){
1260            System.out.println(user);
1261          }
1262        }
1263
1264        @Test
1265        public void test2() {
1266          service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1267          UserVO user = service.getUser("dooly");
1268          System.out.println(user);
1269        }
1270
1271    5)사용자 정보 삭제 테스트
1272      -Users.xml
1273        <delete id="delete" parameterClass="String">
1274          DELETE FROM USERS WHERE  userid = #id#
1275        </delete>
1276
1277      -com.example.dao.UserDaoImplJDBC.java 코드 수정
1278        @Override
1279        public void delete(String id) {
1280          Reader rd = null;
1281          SqlMapClient smc = null;
1282          UserVO userVO = null;
1283          try {
1284            rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1285            smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1286            smc.delete("Users.delete", id);
1287            System.out.println("삭제된 Record with ID = " + id );
1288          } catch (IOException | SQLException e) {
1289            // TODO Auto-generated catch block
1290            e.printStackTrace();
1291          }
1292        }
1293
1294      -MembershipTest.java 수정
1295        @Test
1296        public void test() {
1297          UserVO user = this.service.getUser("jimin");
1298          System.out.println(user);
1299          assertEquals("한지민", user.getName());
1300        }
1301        @Ignore @Test
1302        public void test1() {
1303          this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1304          for(UserVO user : this.service.getUserList()){
1305            System.out.println(user);
1306          }
1307        }
1308
1309        @Ignore @Test
1310        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)준비
    -mvnrepository(https://mvnrepository.com에서 'Mybatis'로 검색
    -MyBatis에서 3.5.1로 들어가서 아래의 코드를 복사해서 pom.xml에 붙여넣기
      <dependency>
        <groupId>org.mybatis</groupId>
        <artifactId>mybatis</artifactId>
        <version>3.5.1</version>
      </dependency>

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

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

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

    -https://github.com/mybatis/mybatis-3/releases
    -mybatis-3.5.1.zip downloads
    -mybatis-3.5.1.pdf 파일 열기

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

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

2)사용자 조회 테스트
  -com.example.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")
    public class UserServiceImpl implements UserService {

      @Autowired
      UserDao userDao2;

  -com.example.dao.UserDaoImplJDBC2.java 생성

    @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 = 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());
        }

  3)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();
        }
      }
```

```
1471        @Override
1472        public List<UserVO> readAll() {
1473          Reader rd = null;
1474          SqlSession session = null;
1475          List<UserVO> userList = null;
1476          try {
1477            rd = Resources.getResourceAsReader("mybatis-config.xml");
1478            session = new SqlSessionFactoryBuilder().build(rd).openSession();
1479            userList = session.selectList("selectAll");
1480          } catch (IOException e) {
1481            e.printStackTrace();
1482          }
1483          return userList;
1484        }
1485
1486      -MembershipTest.java
1487
1488        @Autowired
1489        UserService service;
1490
1491        @Test
1492        public void test() {
1493          UserVO user = this.service.getUser("jimin");
1494          System.out.println(user);
1495          assertEquals("한지민", user.getName());
1496        }
1497        @Test
1498        public void test1() {
1499          this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1500          for(UserVO user : this.service.getUserList()){
1501            System.out.println(user);
1502          }
1503        }
1504
1505    4)사용자 정보 수정 테스트
1506      -mybatis-mapper.xml
1507
1508        <update id="update" parameterType="userVO">
1509          UPDATE USERS SET name = #{name}, gender = #{gender}, city = #{city}
1510          WHERE userid = #{userId}
1511        </update>
1512
1513      -UserDaoImplJDBC2.java
1514
1515        @Override
1516        public void update(UserVO user) {
1517          Reader rd = null;
1518          SqlSession session = null;
1519          UserVO userVO = null;
1520          try {
1521            rd = Resources.getResourceAsReader("mybatis-config.xml");
1522            session = new SqlSessionFactoryBuilder().build(rd).openSession();
1523            session.update("update", user);
1524            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);
        }

  5)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.example.dao.UserDaoImplJDBC.java 복사 후 붙여넣기
    -이름을 UserDaoImplJDBC3.java로

      @Repository("userDao3")    <---변경
      public class UserDaoImpleJDBC implements UserDao {
        private JdbcTemplate jdbcTemplate;    <---변경

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

        class UserMapper implements RowMapper<UserVO> {
```

```
1633            public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
1634              UserVO user = new UserVO();
1635              user.setUserId(rs.getString("userid"));
1636              user.setName(rs.getString("name"));
1637              user.setGender(rs.getString("gender"));
1638              user.setCity(rs.getString("city"));
1639              return user;
1640            }
1641          }
1642
1643          ...
1644          @Override
1645          public UserVO read(String id) {
1646            String SQL = "SELECT * FROM users WHERE userid = ?";
1647            try {
1648              UserVO user = jdbcTemplate.queryForObject(SQL,
1649                   new Object[] { id }, new UserMapper());
1650              return user;
1651            }catch(EmptyResultDataAccessException e){
1652              return null;
1653            }
1654          }
1655
1656      -com.example.service.UserServiceImpl.java 수정
1657
1658          @Service("userService")
1659          public class UserServiceImpl implements UserService {
1660
1661            @Autowired
1662            UserDao userdao3;     <---변경
1663
1664            ...
1665            @Override
1666            public UserVO getUser(String id) {
1667              return userdao3.read(id);
1668            }
1669
1670      -/src/test/java/MembershipTest.java
1671
1672          @Test
1673          public void test() {
1674            //사용자 조회 테스트
1675            UserVO user = service.getUser("jimin");
1676            System.out.println(user);
1677            assertEquals("한지민", user.getName());
1678          }
1679
1680      2)사용자 등록 및 목록 조회 테스트
1681        -com.example.dao.UserDaoImplJDBC.java 코드 수정
1682
1683          @Override
1684          public void insert(UserVO user) {
1685            String SQL = "INSERT INTO users (userid, name, gender,city) VALUES (?, ?, ?, ?)";
1686            jdbcTemplate.update(SQL, user.getUserId(), user.getName(), user.getGender(),
```

```
            user.getCity());
1687
1688        System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" +
            user.getName());
1689      }
1690
1691      @Override
1692      public List<UserVO> readAll() {
1693        String SQL = "SELECT * FROM users";
1694        List<UserVO>  userList = jdbcTemplate.query(SQL, new UserMapper());
1695        return userList;
1696      }
1697
1698    -com.example.service.UserServiceImpl.java 코드 수정
1699
1700      @Override
1701      public void insertUser(UserVO user) {
1702        userDao3.insert(user);
1703      }
1704
1705      @Override
1706      public List<UserVO> getUserList() {
1707        return userDao3.readAll();
1708      }
1709
1710    -/src/test/java/MembershipTest.java
1711
1712      @Test
1713      public void test1() {
1714        //사용자 등록 및 목록조회 테스트
1715        this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1716        for(UserVO user : this.service.getUserList()){
1717          System.out.println(user);
1718        }
1719      }
1720
1721  3)사용자 정보 수정 테스트
1722    -com.example.dao.UserDaoImplJDBC.java 코드 수정
1723
1724      @Override
1725      public void update(UserVO user) {
1726        String SQL = "UPDATE users  SET name = ?, gender = ?, city = ? WHERE userid = ?";
1727        jdbcTemplate.update(SQL, user.getName(), user.getGender(),
            user.getCity(),user.getUserId());
1728        System.out.println("갱신된 Record with ID = " + user.getUserId() );
1729      }
1730
1731    -com.example.service.UserServiceImpl.java 코드 수정
1732
1733      @Override
1734      public void updateUser(UserVO user) {
1735        userDao3.update(user);
1736      }
1737
```

-/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.example.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.example.service.UserServiceImpl.java 코드 수정

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