1. Spring JDBC Architecture
  1)개요
    -대부분의 중/대규모 web application은 효율적인 개발 및 유지 보수를 위하여 계층화(Layering)하여 개발하는 것이 원칙이다.
    -사용자관리 project architecture에서 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 web client의 요청 및 응답을 처리
  2)상위계층(service 계층, data access 계층)에서 발생하는 Exception에 대한 처리
  3)최종 UI에서 표현해야 할 domain model을 사용
  4)최종 UI에서 입력한 data에 대한 유효성 검증(Validation) 기능을 제공
  5)Business logic과 최종 UI를 분리하기 위한 controller 기능을 제공
  6)@Controller annotation을 사용하여 작성된 Controller class가 이 계층에 속함

3. Service Layer
  1)Application business logic 처리와 business와 관련된 domain model의 적합성 검증
  2)Transaction 처리
  3)Presentation Layer와 Data Access Layer 사이를 연결하는 역할로서 두 계층이 직접적으로 통신하지 않게 하여 application의 유연성을 증가
  4)다른 계층들과 통신하기 위한 interface 제공
  5)Service interface와 @Service annotation을 사용하여 작성된 Service 구현 class가 이 계층에 속함.

4. Data Access Layer
  1)영구 저장소(관계형 Database)의 data를 조작하는 data access logic을 객체화
  2)영구 저장소의 data를 조회, 등록, 수정, 삭제함
  3)ORM(Object Relational Mapping) framework(MyBatis, Hibernate)를 주로 사용하는 계층
  4)DAO interface와 @Repository annotation을 사용하여 작성된 DAO 구현 class가 이 계층에 속함.

5. Domain Model Class
  1)관계형 Database의 entity와 비슷한 개념을 가지는 것으로 실제 VO(Value Object) 혹은 DTO(Data Transfer Object) 객체에 해당
  2)Domain model class는 3개의 계층 전체에 걸쳐 사용
  3)private으로 선언된 멤버변수가 있고, 그 변수에 대한 getter와 setter method를 가진 class를 말함.

6. Data access 공통 개념
  1)DAO(Data Access Object) Pattern
    -Data access 계층은 DAO pattern을 적용하여 business logic과 data access logic을 분리하는 것이 원칙이다.
    -Database 접속과 SQL 발행 같은 data access 처리를 DAO라고 불리는 object로 분리하는 pattern이다.

-Business logic이 없거나 단순하면 DAO와 service 계층을 통합할 수도 있지만, 의미 있는 business logic을 가진 enterprise application이라면 data access 계층을 DAO pattern으로 분리해야 한다.
-DAO pattern은 service계층에 영향을 주지 않고 data access 기술을 변경할 수 있는 장점을 가지고 있다.
-Business logic -> Dao interface -> XxxDao(CRUD구현) -> Database
-Dao class에 data access 처리를 기술하겠지만, 그 처리를 구현하는 Java기술은 여러 가지다.
--Dao Interface -> XxxDao(CRUD) -> JDBC/Hibernate/MyBatis(iBATIS)/JPA/JDO등등등 -> Database
--Spring에서는 새로운 data access 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.
--Data access 처리를 간결하게 기술할 수 있다.
--Spring이 제공하는 범용적이고 체계적인 data access 예외를 이용할 수 있다.
--Spring의 transaction 기능을 이용할 수 있다.

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

7. DataSource 구현 class의 종류
1)Test 환경을 위한 DataSource
-SimpleDriverDataSource
--Spring이 제공하는 가장 단순한 DataSource 구현 class
--getConnection()을 호출할 때마다 매번 DB Connection을 새로 만들고 따로 Pool을 관리하지 않으므로 단순한 test용으로만 사용해야 한다.

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

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

8. DataSource 구성하기
1)Spring으로 독립형 application을 개발하는 경우 application context XML file에서 data source를 구성할 수 있다.
2)Enterprise application을 개발하는 경우 application server의 JNDI에 binding되는 data source를 정의한 다음 application에서 사용할 JNDI binding data source를 application context XML file에서 검색할 수 있다.

```
<context:property-placeholder location="classpath:dbinfo.properties" />
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource"
destroy-method="close">
```

```
        <property name="driverClassName" value="${db.driverClass}" />
        <property name="url" value="${db.url}" />
        <property name="username" value="${db.username}" />
        <property name="password" value="${db.password}" />
      </bean>
```

3)Java EE 환경에서 DataSource 구성

-Application server로 배포되는 enterprise application을 개발하는 경우 Spring jee schema의 <jndi-lookup> 요소를 사용해 ApplicationContext의 Spring bean에 JNDI binding data source를 제공할 수 있다.

```
        <jee:jndi-lookup jndi-name="java:comp/env/jdbc/bankAppDb" id="dataSource" />
```

-여기서 jndi-name 속성에는 javax.sql.DataSource 객체를 JNDI에 binding할 JNDI 이름을 지정하며 id 속성에는 javax.sql.DataSource 객체를 ApplicationContext에 등록할 bean 이름을 지정한다.

9. 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 code가 반복적으로 사용되며, DB에 따라 일관성 없는 정보를 가진 채 Checked Exception으로 처리한다.
-장점
  --대부분의 개발자가 잘 알고 있는 친숙한 data access 기술로 별도의 학습 없이 개발이 가능
-단점
  --Connection과 같은 공유 resource를 제대로 release 해주지 않으면 system의 자원이 바닥나는 bug 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 binding 할 parameter를 넘겨주거나, query의 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 source code가 너무 길어지고 또한 connection이나 PreparedStatement를 얻고 나면 반드시 연결 해제를 처리해야 하지만 깜빡 잊어버리는 개발자도 있을 수 있다.
-그래서 연결이 해제되지 않으면 Database의 resource 고갈이나 memory 누수의 원인이 되어 최악의 경우에는 system이 정지할 가능성도 있다.
-Data access 오류시 오류 원인을 특정하고 싶을 때는 SQLException의 오류 code를 가져와 값을 조사할 필요가 있다.
-더욱이 오류 code는 Database 제품마다 값이 다르므로 Database 제품이 바뀌면 다시 수정해야만 한다.
-또한 SQLException은 compile 시 예외 처리 유무를 검사하므로 source code 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 query 실행 결과가 한 건 이상이면 ResultSet loop를 만들어서 반복한다.
-Exception 처리와 반환
--JDBC 작업 중 발생하는 모든 예외는 Spring JDBC 예외 변환기가 처리한다.
--Checked Exception인 SQLException을 Runtime Exception인 DataAccessException type으로 변환
-Transaction 처리
--Transaction과 관련된 모든 작업에 대해서는 신경쓰지 않아도 된다.

5)Spring JDBC의 JdbcTemplate Class
-Spring JDBC가 제공하는 class 중 하나
-JDBC의 모든 기능을 최대한 활용할 수 있는 유연성을 제공하는 class
-Connection, Statement, ResultSet 객체의 관리, JDBC 예외 포착 및 이해하기 쉬운 예외로 변환(예 :IncorrectResultSetColumnCountException 및 CannotGetJdbcConnectionException), 일괄 작업 수행 등을 관리한다.
-Application 개발자는 JdbcTemplate class로 SQL을 제공하고 SQL이 실행된 후 결과를 가져오기만 하면 된다.
-javax.sql.DataSource 객체의 wrapper 역할을 하므로 javax.sql.DataSource 객체를 직접 다룰 필요가 없다.
-일반적으로 JdbcTemplate instance는 연결을 얻을 javax.sql.DataSource 객체에 대한 참조를 사용해 초기화된다.

```
<bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
  <property name="dataSource" ref="dataSource" />
</bean>
<bean id="dataSource" class="org.apache.commons.dbcp.BasicDataSource" ...>
  ...
</bean>
```

-Application에서 JNDI binding data source를 사용하는 경우 jee schema의 <jndi-lookup> 요소를 사용해 JNDI binding data source를 Spring container에 bean으로 등록할 수 있다.
-JdbcTemplate class는 다음 예제와 같이 <jndi-lookup> 요소로 등록한 javax.sql.DataSource bean을 참조할 수 있다.

```
<beans ...
  xmlns:jee="http://www.springframework.org/schema/jee"
    xsi:schemaLocation="...
      http://www.springframework.org/schema/jee
      http://www.springframework.org/schema/jee/spring-jee-4.0.xsd">

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

  <jee:jndi-lookup jndi-name="java:comp/env/jdbc/bankAppDb" id="dataSource" />
    ...
</beans>
```

-실행, 조회, 배치의 3가지 작업 제공
--실행 : Insert나 Update같이 DB의 data에 변경이 일어나는 query를 수행하는 작업

--조회 : Select를 이용해 data를 조회하는 작업
--Batch : 여러 개의 query를 한번에 수행해야 하는 작업

-JdbcTemplate instance는 thread로 부터 안전하다.
-즉, application의 여러 DAO에서 JdbcTemplate class의 동일한 instance를 공유함으로써 Database와 상호작용할 수 있다.

6)JdbcTemplate class 생성
-JdbcTemplate은 DataSource를 parameter로 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 multithread 환경에서도 안전하게 공유해서 쓸 수 있기 때문에 DAO class의 instance 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() method
-INSERT, UPDATE, DELETE와 같은 SQL을 실행할 때 사용.
int update(String sql, [SQL 파라미터])
-이 method를 호출할 때는 SQL과 함께 바인딩 할 파라미터는 Object 타입 가변인자(Object ... args)를 사용할 수 있다.
-이 method의 return값은 SQL 실행으로 영향받은 record의 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() method
-SELECT SQL을 실행하여 하나의 Row를 가져올 때 사용.
<T> T queryForObject(String sql, [SQL parameter], RowMapper<T> rm)
-SQL 실행 결과는 여러 개의 column을 가진 하나의 Row
-T는 VO 객체의 type에 해당
-SQL 실행 결과로 돌아온 여러 개의 column을 가진 한 개의 row를 RowMapper 콜백을 이용해 VO 객체로 mapping한다.
-사용 예

```
234         public User findUser(String id){
235           return this.jdbcTemplate.queryForObject("SELECT * FROM users WHERE id=?",
236             new Object [] {id},
237             new RowMapper<User>(){
238               public User mapRow(ResultSet rs, int rowNum) throws SQLException{
239                 User user = new User();
240                 user.setId(rs.getString("id"));
241                 user.setName(rs.getString("name"));
242                 user.setPassword(rs.getString("password"));
243                 return user;
244               }
245             }
246         }
247
248     9)JdbcTemplate class의 query() method
249       -SELECT SQL을 실행하여 여러 개의 row를 가져올 때 사용.
250
251         <T> List<T> query(String sql, [SQL parameter], RowMapper<T> rm)
252
253       -SQL 실행 결과로 돌아온 여러 개의 column을 가진 여러 개의 row를 RowMapper 콜백을 이용해 VO 객체로
          mapping해준다.
254       -결과 값은 mapping한 VO 객체를 포함하고 있는 List 형태로 받는다.
255       -List의 각 요소가 하나의 row에 해당한다.
256
257
258   10. Spring JDBC 환경설정
259     1)Oracle Jdbc Driver library 검색 및 설치
260
261   ************************************************************************
262     ※Oracle의 경우 어떤 driver를 pom.xml에 넣어도 error가 난다.
263     원래는 Oracle 12C인 경우 Maven Repository에서 'oracle ojdbc8'으로, Oracle 11g인 경우는 'oracle
          ojdbc6'로 검색해야 한다.
264     1)'oracle ojdbc7'으로 검색시 12.1.0.2
265
266       <!-- https://mvnrepository.com/artifact/com.github.noraui/ojdbc7 -->
267       <dependency>
268           <groupId>com.github.noraui</groupId>
269           <artifactId>ojdbc7</artifactId>
270           <version>12.1.0.2</version>
271       </dependency>
272
273     2)'oracle ojdbc6'으로 검색시 11.1.0.7.0
274
275       <!-- https://mvnrepository.com/artifact/com.oracle/ojdbc6 -->
276       <dependency>
277           <groupId>com.oracle</groupId>
278           <artifactId>ojdbc6</artifactId>
279           <version>11.1.0.7.0</version>
280           <scope>test</scope>
281       </dependency>
282
283       하지만 어떤 version도 Maven에서 error가 난다. http://suyou.tistory.com/68 참조.
284       Maven에서 Oracle driver를 찾지 못하는 것은 아마도 저작권 문제로 보인다.
285       그래서 Oracle site에서 직접 driver를 download 받아서 Maven을 이용해서 Maven Local Repository에
```

install을 하고
install된 version으로 pom.xml에 dependency 설정을 해야 한다.

① Oracle homepage에서 Orcle 12C jdbc driver를 download 받는다. -->ojdbc8.jar
② Maven installer를 이용해서 Maven repository에 설치한다.
mvn install:install-file -Dfile="file이름(위치까지)" -DgroupId=그룹아이디 -DartifactId=파일이름 -Dversion=버전 -Dpackaging=jar
위에명령을 cmd에서 실행한다.
자기 version에 맞게 해당항목을 변경한 다음 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] --

install 명령을 실행하면 Maven depository에 해당 driver가 설치된다.
위에서는 C:\Users\webnbiz01\.m2\repository\com\oracle\ojdbc8\12.2 에 설치된 것이다.
해당 directory로 이동하면 jar file과 pom file이 있다.
pom file의 groupId, artifactId, version을 pom.xml에 dependency로 설정하면 된다.

③ pom.xml에 dependency를 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>


11. 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 folder 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 file 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 dependency 추가하고 설치
```

```
390
391         <!-- https://mvnrepository.com/artifact/org.springframework/spring-context -->
392         <dependency>
393             <groupId>org.springframework</groupId>
394             <artifactId>spring-context</artifactId>
395             <version>4.3.24.RELEASE</version>
396         </dependency>
397       -pom.xml에 붙여 넣고 Maven Install 하기
398
399    10)Spring JDBC 설치
400       -JdbcTemplate를 사용하기 위해 pom.xml에 다음 dependency를 추가해야 함.
401
402         <!-- https://mvnrepository.com/artifact/org.springframework/spring-jdbc -->
403         <dependency>
404             <groupId>org.springframework</groupId>
405             <artifactId>spring-jdbc</artifactId>
406             <version>4.3.24.RELEASE</version>
407         </dependency>
408
409       -pom.xml에 붙여 넣고 Maven Install 하기
410         [INFO] BUILD SUCCESS
411
412    11)Bean Configuration XML 작성
413       -/src/config > right-click > New > Other > Spring > Spring Bean Configuration File
414       -File name : beans.xml > Next
415       -Check [beans - http://www.springframework.org/schema/beans]
416       -Check [http://www.springframework.org/schema/beans/spring-beans-4.3.xsd]
417       -Finish
418
419         <?xml version="1.0" encoding="UTF-8"?>
420         <beans xmlns="http://www.springframework.org/schema/beans"
421         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
422         xsi:schemaLocation="http://www.springframework.org/schema/beans
423         http://www.springframework.org/schema/beans/spring-beans.xsd">
424
425         </beans>
426
427       -Namespace tab에서 context - http://www.springframework.org/schema/context check
428
429         <context:property-placeholder location="classpath:dbinfo.properties" />
430         <bean id="dataSource"
          class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
431           <property name="driverClass" value="${db.driverClass}" />
432           <property name="url" value="${db.url}" />
433           <property name="username" value="${db.username}" />
434           <property name="password" value="${db.password}" />
435         </bean>
436
437    12)/src/com.example.UserClient.java 코드 추가
438
439         package com.example;
440
441         import java.sql.SQLException;
442
```

```
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

12. 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 service 계층을 연결하는 역할을 하는 class
---JSP에서 UserController를 통해서 service 계층의 UserService를 사용하게 된다.
---Service 계층의 UserService interface를 구현하나 객체를 IoC 컨테이너가 주입해 준다.

-Service Layer
--UserService<Interface>
---Service 계층에 속한 상위 interface
--UserServiceImpl<Class>
---UserSerive interface를 구현한 class
---복잡한 업무 logic이 있을 경우에는 이 class에서 업무 logic을 구현하면 된다.
---Data access 계층의 userDao interface를 구현한 객체를 IoC container가 주입해준다.

```
497     -Data Access Layer
498       --UserDao<Interface>
499         ---Data access 계층에 속한 상위 interface
500       --UserDaoImplJDBC<Class> - Spring JDBC 구현
501         ---UserDao interface를 구현한 class로 이 class에서는 data access logic을 구현하면 된다.
502         ---Spring JDBC를 사용하는 경우에는 DataSource를 IoC container가 주입해준다.
503         ---MyBatis를 사용하는 경우에는 SqlSession을 IoC container가 주입해준다.
504
505   4)In Package Explorer > right-click > New > Java Project
506     -Project name : Membership
507
508   5)/src > right-click > New > Package
509     -Package name : com.example.vo
510
511   6)com.example.vo.UserVO.java 생성
512
513     package com.example.vo;
514
515     public class UserVO {
516
517       private String userId;
518       private String name;
519       private String gender;
520       private String city;
521
522       public UserVO() {}
523
524       public UserVO(String userId, String name, String gender, String city) {
525         this.userId = userId;
526         this.name = name;
527         this.gender = gender;
528         this.city = city;
529       }
530
531       public String getUserId() {
532         return userId;
533       }
534
535       public void setUserId(String userId) {
536         this.userId = userId;
537       }
538
539       public String getName() {
540         return name;
541       }
542
543       public void setName(String name) {
544         this.name = name;
545       }
546
547       public String getGender() {
548         return gender;
549       }
550
```

```
551         public void setGender(String gender) {
552           this.gender = gender;
553         }
554
555         public String getCity() {
556           return city;
557         }
558
559         public void setCity(String city) {
560           this.city = city;
561         }
562
563         @Override
564         public String toString() {
565           return "User [userId=" + userId + ", name=" + name + ", gender="
566               + gender + ", city=" + city + "]";
567         }
568       }
569
570     7)/src > right-click > New > Package
571       -Package name : com.example.service
572
573     8)UserService interface 생성
574       -com.example.service.UserService.java
575
576       package com.example.service;
577
578       import java.util.List;
579       import com.example.vo.UserVO;
580
581       public interface UserService {
582
583         void insertUser(UserVO user);
584
585         List<UserVO> getUserList();
586
587         void deleteUser(String id);
588
589         UserVO getUser(String id);
590
591         void updateUser(UserVO user);
592       }
593
594     9)UserServiceImpl class 생성
595       -com.example.service.UserServiceImpl.java
596         package com.example.service;
597
598         import java.util.List;
599         import com.example.vo.UserVO;
600
601         public class UserServiceImpl implements UserService {
602
603           @Override
604           public void insertUser(UserVO user) {
```

```
605           // TODO Auto-generated method stub
606
607         }
608
609         @Override
610         public List<UserVO> getUserList() {
611           // TODO Auto-generated method stub
612           return null;
613         }
614
615         @Override
616         public void deleteUser(String id) {
617           // TODO Auto-generated method stub
618
619         }
620
621         @Override
622         public UserVO getUser(String id) {
623           // TODO Auto-generated method stub
624           return null;
625         }
626
627         @Override
628         public void updateUser(UserVO user) {
629           // TODO Auto-generated method stub
630
631         }
632       }
633
634   10)/src > right-click > New > Package
635     -Package name : com.example.dao
636
637   11)UserDao interface 생성
638     -com.example.dao.UserDao.java
639
640       package com.example.dao;
641
642       import java.util.List;
643       import com.example.vo.UserVO;
644
645       public interface UserDao {
646         void insert(UserVO user);
647
648         List<UserVO> readAll();
649
650         void update(UserVO user);
651
652         void delete(String id);
653
654         UserVO read(String id);
655       }
656
657   12)UserDaoImplJDBC class 생성
658     -com.example.dao.UserDaoImplJDBC.java
```

```
659       package com.example.dao;
660
661       import java.util.List;
662       import com.example.vo.UserVO;
663
664       public class UserDaoImplJDBC implements UserDao {
665
666         @Override
667         public void insert(UserVO user) {
668           // TODO Auto-generated method stub
669
670         }
671
672         @Override
673         public List<UserVO> readAll() {
674           // TODO Auto-generated method stub
675           return null;
676         }
677
678         @Override
679         public void update(UserVO user) {
680           // TODO Auto-generated method stub
681
682         }
683
684         @Override
685         public void delete(String id) {
686           // TODO Auto-generated method stub
687
688         }
689
690         @Override
691         public UserVO read(String id) {
692           // TODO Auto-generated method stub
693           return null;
694         }
695       }
696
697   13)Java Project를 Spring Project로 변환
698     -Membership Project > right-click > Configuration > Convert to Maven Project
699       --Project : /Membership
700       --Group Id : Membership
701       --Artifact Id : Membership
702       --version : 0.0.1-SNAPSHOT
703       --Packaging : jar
704       --Finish
705
706       -Membership Project > right-click > Spring > Add Spring Project Nature
707
708       -pom.xml 파일에 Spring Context Dependency 추가하기
709         <version>0.0.1-SNAPSHOT</version>
710         <dependencies>
711           <dependency>
712             <groupId>org.springframework</groupId>
```

```
            <artifactId>spring-context</artifactId>
            <version>4.3.24.RELEASE</version>
          </dependency>
        </dependencies>

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

  14)Oracle Jdbc Driver 설치
    <dependency>
      <groupId>com.oracle</groupId>
      <artifactId>ojdbc8</artifactId>
      <version>12.2</version>
    </dependency>

    <참고>
    -MySQL일 경우에는 'spring mysql'로 검색하여 MySQL Connector/J를 설치한다.
      <dependency>
          <groupId>mysql</groupId>
          <artifactId>mysql-connector-java</artifactId>
          <version>6.0.6</version>
      </dependency>

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

      <dependency>
          <groupId>org.springframework</groupId>
          <artifactId>spring-jdbc</artifactId>
          <version>4.3.24.RELEASE</version>
      </dependency>

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

  16)resource folder 생성
    -Membership project > right-click > Build Path > Coinfigure Build Path
    -Source Tab > Add Folder > Select Membership project > Click [Create New Folder] button
    -Folder name : resources > Finish > OK
    -Apply and Close

  17)dbinfo.properties 파일 생성
    -/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=hr
      db.password=hr

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

```
767         db.username=root
768         db.password=javamysql
769
770     18)Bean Configuration XML 작성
771       -/resources > right-click > New > Spring Bean Configuration File
772       -File name : beans.xml > Finish
773       -Namespace Tab
774       -Check context - http://www.springframework.org/schema/context
775
776         <?xml version="1.0" encoding="UTF-8"?>
777         <beans xmlns="http://www.springframework.org/schema/beans"
778           xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
779           xmlns:context="http://www.springframework.org/schema/context"
780           xsi:schemaLocation="http://www.springframework.org/schema/beans
              http://www.springframework.org/schema/beans/spring-beans.xsd
781             http://www.springframework.org/schema/context
                http://www.springframework.org/schema/context/spring-context-3.2.xsd">
782
783           <context:property-placeholder location="classpath:dbinfo.properties" />
784           <bean id="dataSource"
              class="org.springframework.jdbc.datasource.SimpleDriverDataSource">
785             <property name="driverClass" value="${db.driverClass}" />
786             <property name="url" value="${db.url}" />
787             <property name="username" value="${db.username}" />
788             <property name="password" value="${db.password}" />
789           </bean>
790         </beans>
791
792     19)사용자 관리 project의 Bean 등록 및 의존 관계 설정
793       -<context:component-scan> tag 사용
794       -@Service, @Repository annotation을 선언한 class들과 @Autowired annotation을 선언하여 의존관계
          를 설정한 class들이 위치한 package를 Scan하기 위한 설정을 XML에 해주어야 한다.
795       -beans.xml에 다음 code를 추가한다.
796
797         <context:component-scan base-package="com.example" />
798
799     20)Spring TestContext Framework 사용하기
800       -/src > right-click > New > Package
801       -Package Name : com.example.test
802       -com.example.test > right-click > New > JUnit Test Case
803       -Name : MembershipTest > Finish
804       -Select [Not now] > OK
805
806         package com.example.test;
807
808         import org.junit.Test;
809         import org.junit.runner.RunWith;
810         import org.springframework.beans.factory.annotation.Autowired;
811         import org.springframework.test.context.ContextConfiguration;
812         import org.springframework.test.context.junit4.SpringJUnit4ClassRunner;
813
814         import com.example.service.UserService;
815
816         @RunWith(SpringJUnit4ClassRunner.class)
```

```
817         @ContextConfiguration(locations="classpath:beans.xml")
818         public class MembershipTest {
819
820           @Autowired
821           UserService service;
822
823           @Test
824           public void test() {
825
826           }
827         }
828
829     21)Oracle JDBC Driver(ojdbc) Project BuildPath에 추가
830       -ojdbc6.jar  <--안해도 됨. 왜냐하면 이미 pom.xml에서 추가했기 때문
831
832       <참고>
833       -ojdbc8.jar(http://www.oracle.com/technetwork/database/features/jdbc/jdbc-ucp-122-3110062.html)
834       <참고>
835       -MySQL일 경우에는
          mysql-connector-java-5.1.42-bin.jar(https://dev.mysql.com/downloads/connector/j/) 추가
836
837
838   13. JDBC를 이용한 Membership Project
839     1)사용자 조회 test
840       -com.example.dao.UserDaoImplJDBC.java 수정
841
842         @Repository("userDao")
843         public class UserDaoImplJDBC implements UserDao {
844           private DataSource dataSource;
845
846           @Autowired
847           public void setDataSource(DataSource dataSource) {
848             this.dataSource = dataSource;
849           }
850
851           ...
852           @Override
853           public UserVO read(String id) {
854             Connection conn = null;
855             PreparedStatement pstmt = null;
856             ResultSet rs = null;
857             UserVO userVO = null;
858             try {
859               conn = this.dataSource.getConnection();
860               pstmt = conn.prepareStatement("SELECT * FROM users WHERE userid = ?");
861               pstmt.setString(1, id);
862               rs = pstmt.executeQuery();
863               rs.next();
864               userVO = new UserVO(rs.getString("userid"), rs.getString("name"),
                  rs.getString("gender"), rs.getString("city"));
865             }catch(SQLException ex) {
866               System.out.println(ex);
867             }finally {
```

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

  -com.example.service.UserServiceImpl.java 수정

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

      @Autowired
      UserDao userDao;

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

  -com.example.test.MembershipTest.java

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

  -right-click > Run As > Junit Test
  -결과 -> Junit View에 초록색 bar
    UserVO [userId=jimin, name=한지민, gender=여, city=서울]

2)사용자 등록 및 목록 조회 test
  -com.example.dao.UserDaoImplJDBC.java code 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();
```

```
922             System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" +
                user.getName());
923           }catch(SQLException ex) {
924             System.out.println(ex);
925           }finally {
926             try {
927               if(conn != null) conn.close();
928               if(pstmt != null) pstmt.close();
929             }catch(SQLException ex) {
930               System.out.println(ex);
931             }
932           }
933         }
934
935         @Override
936         public List<UserVO> readAll() {
937           Connection conn = null;
938           Statement stmt = null;
939           ResultSet rs = null;
940           List<UserVO> userList = null;
941           try {
942             conn = this.dataSource.getConnection();
943             stmt = conn.createStatement();
944             rs = stmt.executeQuery("SELECT * FROM users");
945             userList = new ArrayList<UserVO>();
946             while(rs.next()) {
947               UserVO userVO = new UserVO(rs.getString("userid"), rs.getString("name"),
                  rs.getString("gender"), rs.getString("city"));
948               userList.add(userVO);
949             }
950           }catch(SQLException ex) {
951             System.out.println(ex);
952           }finally {
953             try {
954               if(conn != null) conn.close();
955               if(stmt != null) stmt.close();
956               if(rs != null) rs.close();
957             }catch(SQLException ex) {
958               System.out.println(ex);
959             }
960           }
961           return userList;
962         }
963
964       -com.example.service.UserServiceImpl.java code 수정
965
966           @Override
967           public void insertUser(UserVO user) {
968             userDao.insert(user);
969           }
970
971           @Override
972           public List<UserVO> getUserList() {
973             return userDao.readAll();
```

```
        }

    -com.example.test.MembershipTest.java

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

    -right-click > Run As > Junit Test
    -결과 -> Junit View에 초록색 bar
    UserVO [userId=jimin, name=한지민, gender=여, city=서울]
    등록된 Record UserId=dooly Name=둘리
    UserVO [userId=dooly, name=둘리, gender=남, city=경기]
    UserVO [userId=jimin, name=한지민, gender=여, city=서울]


  3)사용자 정보 수정 test
    -com.example.dao.UserDaoImplJDBC.java code 수정

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

        @Override
```

```
1028          public void updateUser(UserVO user) {
1029            userDao.update(user);
1030          }
1031
1032      -com.example.test.MembershipTest.java
1033
1034          @Ignore @Test
1035          public void test1() {
1036            //사용자 등록 및 목록조회 test
1037            this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1038            for(UserVO user : this.service.getUserList()){
1039              System.out.println(user);
1040            }
1041          }
1042
1043          @Test
1044          public void test2() {
1045            //사용자 정보 수정 test
1046            service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1047            UserVO user = service.getUser("dooly");
1048            System.out.println(user);
1049          }
1050
1051      -right-click > Run As > Junit Test
1052      -결과 -> Junit View에 초록색 bar
1053      UserVO [userId=jimin, name=한지민, gender=여, city=서울]
1054      갱신된 Record with ID = dooly
1055      UserVO [userId=dooly, name=김둘리, gender=여, city=부산]
1056
1057    4)사용자 정보 삭제 test
1058      -com.example.dao.UserDaoImplJDBC.java code 수정
1059
1060        @Override
1061        public void delete(String id) {
1062          Connection conn = null;
1063          PreparedStatement pstmt = null;
1064          try {
1065            conn = this.dataSource.getConnection();
1066            pstmt = conn.prepareStatement("DELETE FROM users WHERE userid = ?");
1067            pstmt.setString(1, id);
1068            pstmt.executeUpdate();
1069            System.out.println("삭제된 Record with ID = " + id );
1070          }catch(SQLException ex) {
1071            System.out.println(ex);
1072          }finally {
1073            try {
1074              if(conn != null) conn.close();
1075              if(pstmt != null) pstmt.close();
1076            }catch(SQLException ex) {
1077              System.out.println(ex);
1078            }
1079          }
1080        }
1081
```

-com.example.service.UserServiceImpl.java code 수정

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

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

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

14. iBATIS를 이용한 Membership Project

1)준비

-mvnrepository(https://mvnrepository.com에서 'ibatis'로 검색
-Ibatis Sqlmap에서 2.3.4.726으로 들어가서 아래의 code를 복사해서 pom.xml에 넣기

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

```
1136
1137      -pom.xml에 붙여 넣고 Maven Install 하기
1138        [INFO] BUILD SUCCESS 확인
1139
1140      -SqlMapConfig.xml 생성
1141        --src > right-click > New > Other > XML > XML File > Next
1142        --File name : SqlMapConfig.xml > Finish
1143        --<!DOCTYPE element는 internet에서 sqlmapconfig.xml로 검색
1144
1145        <?xml version="1.0" encoding="UTF-8"?>
1146        <!DOCTYPE sqlMapConfig
1147            PUBLIC "-//ibatis.apache.org//DTD SQL Map Config 2.0//EN"
1148              "http://ibatis.apache.org/dtd/sql-map-config-2.dtd">
1149        <sqlMapConfig>
1150          <properties resource="dbinfo.properties" />
1151          <settings useStatementNamespaces="true"/>
1152          <transactionManager type="JDBC">
1153            <dataSource type="SIMPLE">
1154              <property name="JDBC.Driver" value="${db.driverClass}"/>
1155              <property name="JDBC.ConnectionURL" value="${db.url}"/>
1156              <property name="JDBC.Username" value="${db.username}"/>
1157              <property name="JDBC.Password" value="${db.password}"/>
1158            </dataSource>
1159          </transactionManager>
1160          <sqlMap resource="com/example/dao/Users.xml"/>
1161        </sqlMapConfig>
1162
1163    2)사용자 조회 test
1164      -com.example.dao/Users.xml 생성
1165        <?xml version="1.0" encoding="UTF-8"?>
1166        <!DOCTYPE sqlMap
1167              PUBLIC "-//ibatis.apache.org//DTD SQL Map 2.0//EN"
1168                "http://ibatis.apache.org/dtd/sql-map-2.dtd">
1169        <sqlMap namespace="Users">
1170          <typeAlias alias="userVO" type="com.example.vo.UserVO"/>
1171          <resultMap id="result" class="userVO">
1172            <result property="userId" column="userid"/>
1173            <result property="name" column="name"/>
1174            <result property="gender" column="gender"/>
1175            <result property="city" column="city"/>
1176          </resultMap>
1177          <select id="useResultMap" resultMap="result">
1178            SELECT * FROM users WHERE userid=#id#
1179          </select>
1180        </sqlMap>
1181
1182      -com.example.dao.UserDaoImplJDBC1.java 생성
1183        @Repository("userDao1")
1184        public class UserDaoImplJDBC1 implements UserDao {
1185          @Override
1186          public UserVO read(String id) {
1187            Reader rd = null;
1188            SqlMapClient smc = null;
1189            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 test
          UserVO user = service.getUser("jimin");
          System.out.println(user);
          assertEquals("한지민", user.getName());
        }

  3)사용자 등록 및 목록 조회 test
    -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 {
```

```
1244            rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1245            smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1246            smc.insert("Users.insert", user);
1247            System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" +
                user.getName());
1248          } catch (IOException | SQLException e) {
1249            // TODO Auto-generated catch block
1250            e.printStackTrace();
1251          }
1252        }
1253
1254        @Override
1255        public List<UserVO> readAll() {
1256          Reader rd = null;
1257          SqlMapClient smc = null;
1258          List<UserVO> userList = null;
1259          try {
1260            rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1261            smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1262            userList = (List<UserVO>)smc.queryForList("Users.getAll", null);
1263          } catch (IOException | SQLException e) {
1264            // TODO Auto-generated catch block
1265            e.printStackTrace();
1266          }
1267          return userList;
1268        }
1269
1270      -MembershipTest.java
1271        @Test
1272        public void test1() {
1273          this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1274          for(UserVO user : this.service.getUserList()){
1275            System.out.println(user);
1276          }
1277        }
1278
1279    4)사용자 정보 수정 test
1280      -Users.xml
1281        <update id="update" parameterClass="userVO">
1282          UPDATE USERS
1283          SET    name = #name#, gender = #gender#, city = #city#
1284          WHERE  userId = #userId#
1285        </update>
1286
1287      -com.example.dao.UserDaoImplJDBC1.java code 수정
1288        @Override
1289        public void update(UserVO user) {
1290          Reader rd = null;
1291          SqlMapClient smc = null;
1292          UserVO userVO = null;
1293          try {
1294            rd = Resources.getResourceAsReader("SqlMapConfig.xml");
1295            smc = SqlMapClientBuilder.buildSqlMapClient(rd);
1296            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);
        }

  5)사용자 정보 삭제 test
    -Users.xml
      <delete id="delete" parameterClass="String">
        DELETE FROM USERS WHERE  userid = #id#
      </delete>

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

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

```
1351        public void test1() {
1352          this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1353          for(UserVO user : this.service.getUserList()){
1354            System.out.println(user);
1355          }
1356        }
1357
1358        @Ignore @Test
1359        public void test2() {
1360          service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1361          UserVO user = service.getUser("dooly");
1362          System.out.println(user);
1363        }
1364
1365        @Test
1366        public void test3() {
1367          //사용자 정보 삭제 test
1368          service.deleteUser("dooly");
1369          for(UserVO user : service.getUserList()){
1370            System.out.println(user);
1371          }
1372        }
1373
1374
1375  15. MyBatis를 이용한 Membership Project
1376    1)준비
1377      -mvnrepository(https://mvnrepository.com에서 'Mybatis'로 검색
1378      -MyBatis에서 3.5.1로 들어가서 아래의 code를 복사해서 pom.xml에 붙여넣기
1379        <dependency>
1380          <groupId>org.mybatis</groupId>
1381          <artifactId>mybatis</artifactId>
1382          <version>3.5.1</version>
1383        </dependency>
1384
1385      -pom.xml에 붙여 넣고 Maven Install 하기
1386        [INFO] BUILD SUCCESS 확인
1387
1388      -src/main/resources/dbinfo.properties
1389        db.driverClass=oracle.jdbc.driver.OracleDriver
1390        db.url=jdbc:oracle:thin:@localhost:1521:XE
1391        db.username=hr
1392        db.password=hr
1393
1394      -mybatis-config.xml 생성
1395        --src > right-click > New > Other > XML > XML File > Next
1396        --File name : mybatis-config.xml > Finish
1397
1398      -https://github.com/mybatis/mybatis-3/releases
1399      -mybatis-3.5.1.zip downloads
1400      -mybatis-3.5.1.pdf file 열기
1401
1402        <?xml version="1.0" encoding="UTF-8"?>
1403        <!DOCTYPE configuration
1404              PUBLIC "-//mybatis.org//DTD Config 3.0//EN"
```

```
1405                "http://mybatis.org/dtd/mybatis-3-config.dtd">
1406        <configuration>
1407          <properties resource="dbinfo.properties" />
1408          <typeAliases>
1409            <typeAlias type="com.example.vo.UserVO" alias="userVO" />
1410          </typeAliases>
1411          <environments default="development">
1412            <environment id="development">
1413              <transactionManager type="JDBC"/>
1414              <dataSource type="POOLED">
1415                <property name="driver" value="${db.driverClass}"/>
1416                <property name="url" value="${db.url}"/>
1417                <property name="username" value="${db.username}"/>
1418                <property name="password" value="${db.password}"/>
1419              </dataSource>
1420            </environment>
1421          </environments>
1422          <mappers>
1423            <mapper resource="com/example/dao/mybatis-mapper.xml"/>
1424          </mappers>
1425        </configuration>
1426
1427    2)사용자 조회 test
1428      -com.example.dao/mybatis-mapper.xml
1429
1430        <?xml version="1.0" encoding="UTF-8"?>
1431        <!DOCTYPE mapper
1432                PUBLIC "-//mybatis.org//DTD Mapper 3.0//EN"
1433                  "http://mybatis.org/dtd/mybatis-3-mapper.dtd">
1434        <mapper namespace="com.example.vo.UserVO">
1435          <resultMap id="userVOResult" type="userVO">
1436            <result property="userId" column="userid" />
1437            <result property="name" column="name" />
1438            <result property="gender" column="gender" />
1439            <result property="city" column="city" />
1440          </resultMap>
1441          <select id="select" parameterType="String" resultType="userVO"
              resultMap="userVOResult">
1442            SELECT * FROM USERS WHERE userid = #{id}
1443          </select>
1444        </mapper>
1445
1446      -UserServiceImpl.java 수정
1447        @Service("userService")
1448        public class UserServiceImpl implements UserService {
1449
1450          @Autowired
1451          UserDao userDao2;
1452
1453      -com.example.dao.UserDaoImplJDBC2.java 생성
1454
1455        @Repository("userDao2")
1456        public class UserDaoImplJDBC2 implements UserDao {
1457          ...
```

```
1458          @Override
1459          public UserVO read(String id) {
1460            Reader rd = null;
1461            SqlSession session = null;
1462            UserVO userVO = null;
1463            try {
1464              rd = Resources.getResourceAsReader("mybatis-config.xml");
1465              session = new SqlSessionFactoryBuilder().build(rd).openSession();
1466              userVO = (UserVO)session.selectOne("select", id);
1467            } catch (IOException e) {
1468              // TODO Auto-generated catch block
1469              e.printStackTrace();
1470            }
1471            return userVO;
1472          }
1473
1474      -MembershipTest.java
1475
1476        @RunWith(SpringJUnit4ClassRunner.class)
1477        @ContextConfiguration(locations="classpath:beans.xml")
1478        public class MembershipTest {
1479
1480          @Autowired
1481          UserService service;
1482
1483          @Test
1484          public void test() {
1485            UserVO user = this.service.getUser("jimin");
1486            System.out.println(user);
1487            assertEquals("한지민", user.getName());
1488          }
1489
1490      3)사용자 등록 및 목록 조회 test
1491        -mybatis-mapper.xml
1492
1493          <insert id="insert" parameterType="userVO">
1494            INSERT INTO USERS(userid, name, gender, city)
1495            VALUES (#{userId}, #{name}, #{gender}, #{city})
1496          </insert>
1497
1498          <select id="selectAll" resultType="userVO" resultMap="userVOResult">
1499            SELECT * FROM USERS
1500          </select>
1501
1502        -UserDaoImplJDBC2.java
1503
1504          @Override
1505          public void insert(UserVO user) {
1506            Reader rd = null;
1507            SqlSession session = null;
1508            UserVO userVO = null;
1509            try {
1510              rd = Resources.getResourceAsReader("mybatis-config.xml");
1511              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);
          }
        }

  4)사용자 정보 수정 test
    -mybatis-mapper.xml

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

    -UserDaoImplJDBC2.java

        @Override
```

```
1565        public void update(UserVO user) {
1566          Reader rd = null;
1567          SqlSession session = null;
1568          UserVO userVO = null;
1569          try {
1570            rd = Resources.getResourceAsReader("mybatis-config.xml");
1571            session = new SqlSessionFactoryBuilder().build(rd).openSession();
1572            session.update("update", user);
1573            session.commit();
1574            System.out.println("갱신된 Record with ID = " + user.getUserId() );
1575          } catch (IOException e) {
1576            e.printStackTrace();
1577          }
1578        }
1579
1580      -MembershipTest.java
1581
1582        @Autowired
1583        UserService service;
1584
1585        @Test
1586        public void test() {
1587          UserVO user = this.service.getUser("jimin");
1588          System.out.println(user);
1589          assertEquals("한지민", user.getName());
1590        }
1591        @Ignore @Test
1592        public void test1() {
1593          this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1594          for(UserVO user : this.service.getUserList()){
1595            System.out.println(user);
1596          }
1597        }
1598
1599        @Test
1600        public void test2() {
1601          service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1602          UserVO user = service.getUser("dooly");
1603          System.out.println(user);
1604        }
1605
1606    5)사용자 정보 삭제 test
1607      -mybatis-mapper.xml
1608
1609        <delete id="delete" parameterType="String">
1610          DELETE FROM USERS WHERE  userid = #{id}
1611        </delete>
1612
1613      -UserDaoImplJDBC2.java
1614
1615        @Override
1616        public void delete(String id) {
1617          Reader rd = null;
1618          SqlSession session = null;
```

```
1619          UserVO userVO = null;
1620          try {
1621            rd = Resources.getResourceAsReader("mybatis-config.xml");
1622            session = new SqlSessionFactoryBuilder().build(rd).openSession();
1623            session.delete("delete", id);
1624            session.commit();
1625            System.out.println("삭제된 Record with ID = " + id );
1626          } catch (IOException e) {
1627            e.printStackTrace();
1628          }
1629        }
1630
1631      -MembershipTest.java
1632
1633        @Autowired
1634        UserService service;
1635
1636        @Test
1637        public void test() {
1638          UserVO user = this.service.getUser("jimin");
1639          System.out.println(user);
1640          assertEquals("한지민", user.getName());
1641        }
1642        @Ignore @Test
1643        public void test1() {
1644          this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1645          for(UserVO user : this.service.getUserList()){
1646            System.out.println(user);
1647          }
1648        }
1649
1650        @Ignore @Test
1651        public void test2() {
1652          service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1653          UserVO user = service.getUser("dooly");
1654          System.out.println(user);
1655        }
1656
1657        @Test
1658        public void test3() {
1659          //사용자 정보 삭제 test
1660          service.deleteUser("dooly");
1661          for(UserVO user : service.getUserList()){
1662            System.out.println(user);
1663          }
1664        }
1665
1666
1667  16. JDBCTemplate를 이용한 Membership Project
1668    1)사용자 조회 test
1669      -com.example.dao.UserDaoImplJDBC.java 복사 후 붙여넣기
1670      -이름을 UserDaoImplJDBC3.java로
1671
1672        @Repository("userDao3")    <---변경
```

```
1673        public class UserDaoImpleJDBC implements UserDao {
1674          private JdbcTemplate jdbcTemplate;    <---변경
1675
1676          @Autowired
1677          public void setDataSource(DataSource dataSource) {
1678            this.jdbcTemplate = new JdbcTemplate(dataSource);
1679          }
1680
1681          class UserMapper implements RowMapper<UserVO> {
1682            public UserVO mapRow(ResultSet rs, int rowNum) throws SQLException {
1683              UserVO user = new UserVO();
1684              user.setUserId(rs.getString("userid"));
1685              user.setName(rs.getString("name"));
1686              user.setGender(rs.getString("gender"));
1687              user.setCity(rs.getString("city"));
1688              return user;
1689            }
1690          }
1691
1692          ...
1693          @Override
1694          public UserVO read(String id) {
1695            String SQL = "SELECT * FROM users WHERE userid = ?";
1696            try {
1697              UserVO user = jdbcTemplate.queryForObject(SQL,
1698                   new Object[] { id }, new UserMapper());
1699              return user;
1700            }catch(EmptyResultDataAccessException e){
1701              return null;
1702            }
1703          }
1704
1705      -com.example.service.UserServiceImpl.java 수정
1706
1707          @Service("userService")
1708          public class UserServiceImpl implements UserService {
1709
1710            @Autowired
1711            UserDao userdao3;     <---변경
1712
1713            ...
1714            @Override
1715            public UserVO getUser(String id) {
1716              return userdao3.read(id);
1717            }
1718
1719      -/src/test/java/MembershipTest.java
1720
1721          @Test
1722          public void test() {
1723            //사용자 조회 test
1724            UserVO user = service.getUser("jimin");
1725            System.out.println(user);
1726            assertEquals("한지민", user.getName());
```

```
1727        }
1728
1729    2)사용자 등록 및 목록 조회 test
1730      -com.example.dao.UserDaoImplJDBC.java code 수정
1731
1732        @Override
1733        public void insert(UserVO user) {
1734          String SQL = "INSERT INTO users (userid, name, gender,city) VALUES (?, ?, ?, ?)";
1735          jdbcTemplate.update(SQL, user.getUserId(), user.getName(), user.getGender(),
              user.getCity());
1736
1737          System.out.println("등록된 Record UserId=" + user.getUserId() + " Name=" +
              user.getName());
1738        }
1739
1740        @Override
1741        public List<UserVO> readAll() {
1742          String SQL = "SELECT * FROM users";
1743          List<UserVO>  userList = jdbcTemplate.query(SQL, new UserMapper());
1744          return userList;
1745        }
1746
1747      -com.example.service.UserServiceImpl.java code 수정
1748
1749        @Override
1750        public void insertUser(UserVO user) {
1751          userDao3.insert(user);
1752        }
1753
1754        @Override
1755        public List<UserVO> getUserList() {
1756          return userDao3.readAll();
1757        }
1758
1759      -/src/test/java/MembershipTest.java
1760
1761        @Test
1762        public void test1() {
1763          //사용자 등록 및 목록조회 test
1764          this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1765          for(UserVO user : this.service.getUserList()){
1766            System.out.println(user);
1767          }
1768        }
1769
1770    3)사용자 정보 수정 test
1771      -com.example.dao.UserDaoImplJDBC.java code 수정
1772
1773        @Override
1774        public void update(UserVO user) {
1775          String SQL = "UPDATE users  SET name = ?, gender = ?, city = ? WHERE userid = ?";
1776          jdbcTemplate.update(SQL, user.getName(), user.getGender(),
              user.getCity(),user.getUserId());
1777          System.out.println("갱신된 Record with ID = " + user.getUserId() );
```

```
1778          }
1779
1780      -com.example.service.UserServiceImpl.java code 수정
1781
1782          @Override
1783          public void updateUser(UserVO user) {
1784            userDao3.update(user);
1785          }
1786
1787      -/src/test/java/MembershipTest.java
1788
1789          @Ignore @Test
1790          public void test1() {
1791            //사용자 등록 및 목록조회 test
1792            this.service.insertUser(new UserVO("dooly", "둘리", "남", "경기"));
1793            for(UserVO user : this.service.getUserList()){
1794              System.out.println(user);
1795            }
1796          }
1797
1798          @Test
1799          public void test2() {
1800            //사용자 정보 수정 test
1801            service.updateUser(new UserVO("dooly", "김둘리", "여", "부산"));
1802            UserVO user = service.getUser("dooly");
1803            System.out.println(user);
1804          }
1805
1806    4)사용자 정보 삭제 test
1807      -com.example.dao.UserDaoImplJDBC.java code 수정
1808
1809        @Override
1810        public void delete(String id) {
1811          String SQL = "DELETE FROM users WHERE userid = ?";
1812          jdbcTemplate.update(SQL, id);
1813          System.out.println("삭제된 Record with ID = " + id );
1814        }
1815
1816      -com.example.service.UserServiceImpl.java 코드 수정
1817
1818        @Override
1819        public void deleteUser(String id) {
1820          userDao3.delete(id);
1821        }
1822
1823      -/src/test/java/MembershipTest.java
1824
1825          @Test
1826          public void test3() {
1827            //사용자 정보 삭제 test
1828            service.deleteUser("dooly");
1829            for(UserVO user : service.getUserList()){
1830              System.out.println(user);
1831            }
```

```
1832        }
```