

```
1  HOL : Using JdbcTemplate
2  -----
3  Task1. CRUD of Spring JdbcTemplate
4  1. Create Table
5
6      CREATE TABLE Student(
7          id   INT   NOT NULL AUTO_INCREMENT,
8          name VARCHAR(20) NOT NULL,
9          age  INT   NOT NULL,
10         PRIMARY KEY(id)
11     );
12
13
14  2. In Package Explorer > right-click > New > Java Project
15      1)Project Name : JdbcTemplateDemo
16      2)JRE
17          -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
18      3)Uncheck [Create module-info.java file]
19      4)Next
20      5)Finish
21
22
23  3. src > right-click > New > Package
24      1)Name : com.example
25      2)Finish
26
27
28  4. Java Project를 Spring Project로 변환
29      1)JdbcTemplateDemo Project > right-click > Configure > Convert to Maven Project
30          -Project : /JdbcTemplateDemo
31          -Group Id : JdbcTemplateDemo
32          -Artifact Id : JdbcTemplateDemo
33          -version : 0.0.1-SNAPSHOT
34          -Packaging : jar
35          -Finish
36
37      2)JdbcTemplateDemo Project > right-click > Spring > Add Spring Project Nature
38
39      3)pom.xml file에 Spring Context Dependency 추가하기
40          <version>0.0.1-SNAPSHOT</version>
41          <dependencies>
42              <dependency>
43                  <groupId>org.springframework</groupId>
44                  <artifactId>spring-context</artifactId>
45                  <version>5.3.10</version>
46              </dependency>
47          </dependencies>
48
49      4)pom.xml > right-click > Run As > Maven install
50          [INFO] BUILD SUCCESS 확인
51
52
53  5. Lombok library 추가
54      1)https://mvnrepository.com/에서 'lombok'으로 검색
55      2)'Project Lombok' click
56      3)1.18.20 click
57      4)dependency copy해서 pom.xml에 붙여넣기
58
59          <dependencies>
60              <dependency>
61                  <groupId>org.springframework</groupId>
62                  <artifactId>spring-context</artifactId>
63                  <version>5.3.10</version>
64              </dependency>
65              <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
66              <dependency>
67                  <groupId>org.projectlombok</groupId>
```

```
68         <artifactId>lombok</artifactId>
69         <version>1.18.20</version>
70         <scope>provided</scope>
71     </dependency>
72 </dependencies>
73
```

74 5) pom.xml > right-click > Run As > Maven install
75 [INFO] BUILD SUCCESS 확인
76
77

78 6. pom.xml에 Jdbc Driver 설정하기

79 1) Oracle 12C 이후 version일 경우, mvnrepository에서 oracle로 검색후, Ojdbc8 설치

```
80     <dependency>
81         <groupId>com.oracle.database.jdbc</groupId>
82         <artifactId>ojdbc8</artifactId>
83         <version>21.3.0.0</version>
84     </dependency>
85
```

86 2) Oracle 11g version일 경우

87 -pom.xml에 붙여 넣고 Maven Install 하기

```
88     <dependency>
89         <groupId>com.oracle</groupId>
90         <artifactId>ojdbc6</artifactId>
91         <version>11.2</version>
92     </dependency>
93
```

94 3) MySQL의 경우, MySQL Connector/J로 들어가서

```
95     <dependency>
96         <groupId>mysql</groupId>
97         <artifactId>mysql-connector-java</artifactId>
98         <version>8.0.26</version>
99     </dependency>
100
```

101 4) MariaDB의 경우, MariaDB Java Client로 들어가서

```
102     <dependency>
103         <groupId>org.mariadb.jdbc</groupId>
104         <artifactId>mariadb-java-client</artifactId>
105         <version>2.7.4</version>
106     </dependency>
107
```

108 5) pom.xml > right-click > Run As > Maven install
109 [INFO] BUILD SUCCESS 확인
110
111

112 7. Spring JDBC pom.xml에 추가하기

113 1) pom.xml에 다음 코드 추가

```
114     <dependency>
115         <groupId>org.springframework</groupId>
116         <artifactId>spring-jdbc</artifactId>
117         <version>5.3.10</version>
118     </dependency>
119
```

120
121 2) pom.xml > right-click > Run As > Maven install
122 [INFO] BUILD SUCCESS 확인
123
124

125 8. Student class 생성

126 1) com.example > right-click > New > Class

127 2) Name : Student

128 3) Finish

```
129  
130     package com.example;
131  
132     import lombok.AllArgsConstructor;
133     import lombok.Getter;
134     import lombok.NoArgsConstructor;
```

```
135     import lombok.Setter;
136
137     @Getter
138     @Setter
139     @AllArgsConstructor
140     @NoArgsConstructor
141     public class Student {
142         private int id;
143         private int age;
144         private String name;
145     }
146
147
```

148 9. StudentDao interface 생성

- 149 1)com.example > right-click > New > Interface
- 150 2)Name : StudentDao
- 151 3)Finish

```
152
153     package com.example;
154
155     import java.util.List;
156     import javax.sql.DataSource;
157
158     public class StudentDao {
159         void setDataSource(DataSource ds);
160         void create(String name, int age);
161         void delete(int id);
162         void update(int id, int age);
163         Student getStudent(int id);
164         List<Student> listStudents();
165     }
166
167
```

168 10. resources folder 생성하기

- 169 1)JdbcTemplateDemo project > right-click > New > Source Folder
- 170 2)Folder name : resources
- 171 3)Finish

174 11. resources/dbinfo.properties file 생성

- 175 1)Oracle Database 일 경우
- 176 db.driverClass=oracle.jdbc.driver.OracleDriver
- 177 db.url=jdbc:oracle:thin:@localhost:1521:XE
- 178 db.username=hr
- 179 db.password=hr

181 2)MySQL Database 인 경우

- 182 db.driverClass=com.mysql.jdbc.Driver
- 183 db.url=jdbc:mysql://localhost:3306/world
- 184 db.username=root
- 185 db.password=javamysql

188 12. src/com.example.StudentMapper class 생성

- 189 1)com.example > right-click > New > Class
- 190 2)Name : StudentMapper
- 191 3)Finish

```
192
193     package com.example;
194
195     import java.sql.ResultSet;
196     import java.sql.SQLException;
197
198     import org.springframework.jdbc.core.RowMapper;
199
200     public class StudentMapper implements RowMapper<Student>{
201         public Student mapRow(ResultSet rs, int rowNum) throws SQLException{
```

```

202         Student student = new Student();
203         student.setId(rs.getInt("id"));
204         student.setName(rs.getString("name"));
205         student.setAge(rs.getInt("age"));
206         return student;
207     }
208 }

```

211 13. StudentJDBCTemplate class 생성

212 1)com.example > right-click > New > Class

213 2)Name : StudentJDBCTemplate

214 3)Finish

```

215
216     package com.example;
217
218     import java.util.List;
219     import javax.sql.DataSource;
220
221     public class StudentJDBCTemplate implements StudentDao {
222         private DataSource dataSource;
223         private JdbcTemplate jdbcTemplate;
224
225         @Override
226         public void setDataSource(DataSource ds){
227             this.dataSource = ds
228             this.jdbcTemplate = new JdbcTemplate(dataSource);
229         }
230
231         @Override
232         public void create(String name, int age){
233             String sql = "INSERT INTO Student(name, age) VALUES (?, ?)";
234             this.jdbcTemplate.update(sql, name, age);
235             System.out.println("Insert Success");
236         }
237
238         @Override
239         public Student getStudent(int id){
240             String sql = "SELECT * FROM Student where id = ?";
241             Student student = this.jdbcTemplate.queryForObject(sql, new StudentMapper(), id);
242             return student;
243         }
244
245         @Override
246         public List<Student> listStudents(){
247             String sql = "SELECT * FROM Student";
248             List<Student> students = this.jdbcTemplate.query(sql, new StudentMapper());
249             return students;
250         }
251
252         @Override
253         public void delete(int id){
254             String sql = "DELETE FROM Student WHERE id = ?";
255             this.jdbcTemplate.update(sql, id);
256             System.out.println("Delete Success");
257         }
258
259         @Override
260         public void updae(int id, int age){
261             String sql = "UPDATE Student set age = ? WHERE id = ?";
262             this.jdbcTemplate.update(sql, age, id);
263             System.out.println("Update Success");
264         }
265     }

```

268 14. beans.xml 생성

- 1)resources > right-click > New > Other > Spring > Spring Bean Configuration File > Next
- 2)File name : beans.xml
- 3)Finish
- 4)Namespaces tab > context check

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

  <bean id="dataSource"
    class="org.springframework.jdbc.datasource.DriverManagerDataSource">
    <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>

  <bean id="studentJdbcTemplate" class="com.example.StudentJDBCTemplate">
    <property name="dataSource" ref="dataSource" />
  </bean>
</beans>
```

15. MainClass Class 생성

- 1)com.example > right-click > New > Class
- 2)Name : MainClass
- 3)Finish

```
package com.example;

import java.util.List;
import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainClass {
    public static void main(String[] args) {
        ApplicationContext ctx=new ClassPathXmlApplicationContext("beans.xml");

        StudentJDBCTemplate temp =
            (StudentJDBCTemplate)ctx.getBean("studentJDBCTemplate");

        System.out.println("-----Records Creation-----");
        temp.create("Zara", 11);
        temp.create("Nuha", 2);
        temp.create("Ayan", 15);

        System.out.println("-----Listing Multiple Records-----");
        List<Student> students = temp.listStudents();

        for(Student student : students){
            System.out.print("ID : " + student.getId());
            System.out.print(", Name : " + student.getName());
            System.out.println(", Age : " + student.getAge());
        }

        System.out.println("-----Delete Record with ID = 1 -----");
        temp.delete(1);

        System.out.println("-----Updating Record with ID = 2 -----");
        temp.update(2, 20);

        System.out.println("-----Listing Multiple Records-----");
        List<Student> students = temp.listStudents();
```

```

333         for(Student student : students){
334             System.out.print("ID : " + student.getId());
335             System.out.print(", Name : " + student.getName());
336             System.out.println(", Age : " + student.getAge());
337         }
338     }
339 }

```

342 -----

343 Task2. Example of Spring JdbcTemplate

344 1. Create Table

```

345 CREATE TABLE Employee(
346     id NUMBER(10),
347     name VARCHAR2(100),
348     salary NUMBER(10)
349 );
350

```

353 2. In Package Explorer > right-click > New > Java Project

- 354 1)Project Name : JdbcTemplateDemo1
- 355 2)JRE
 - 356 -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
- 357 3)Uncheck [Create module-info.java file]
- 358 4)Next
- 359 5)Finish

362 3. src > right-click > New > Package

- 363 1)Name : com.example
- 364 2)Finish

367 4. Java Project를 Spring Project로 변환

- 368 1)JdbcTemplateDemo1 Project > right-click > Configure > Convert to Maven Project
 - 369 -Project : /JdbcTemplateDemo1
 - 370 -Group Id : JdbcTemplateDemo1
 - 371 -Artifact Id : JdbcTemplateDemo1
 - 372 -version : 0.0.1-SNAPSHOT
 - 373 -Packaging : jar
 - 374 -Finish

376 2)JdbcTemplateDemo1 Project > right-click > Spring > Add Spring Project Nature

378 3)pom.xml file에 Spring Context Dependency 추가하기

```

379 <version>0.0.1-SNAPSHOT</version>
380 <dependencies>
381     <dependency>
382         <groupId>org.springframework</groupId>
383         <artifactId>spring-context</artifactId>
384         <version>5.3.10</version>
385     </dependency>
386 </dependencies>

```

388 4)pom.xml > right-click > Run As > Maven install

389 [INFO] BUILD SUCCESS 확인

392 5. Lombok library 추가

- 393 1)<https://mvnrepository.com/>에서 'lombok'으로 검색
- 394 2)'Project Lombok' click
- 395 3)1.18.20 click
- 396 4)dependency copy해서 pom.xml에 붙여넣기

```

397
398 <dependencies>
399     <dependency>

```

```
400     <groupId>org.springframework</groupId>
401     <artifactId>spring-context</artifactId>
402     <version>5.3.10</version>
403 </dependency>
404 <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
405 <dependency>
406     <groupId>org.projectlombok</groupId>
407     <artifactId>lombok</artifactId>
408     <version>1.18.20</version>
409     <scope>provided</scope>
410 </dependency>
411 </dependencies>
```

412
413 5) pom.xml > right-click > Run As > Maven install
414 [INFO] BUILD SUCCESS 확인
415
416

417 6. pom.xml에 Jdbc Driver 설정하기

418 1) Oracle 12C 이후 version일 경우, mvnrepository에서 oralc로 검색후, Ojdbc8 설치

```
419 <dependency>
420     <groupId>com.oracle.database.jdbc</groupId>
421     <artifactId>ojdbc8</artifactId>
422     <version>21.3.0.0</version>
423 </dependency>
```

424
425 2) Oracle 11g version일 경우

426 - pom.xml에 붙여 넣고 Maven Install 하기

```
427 <dependency>
428     <groupId>com.oracle</groupId>
429     <artifactId>ojdbc6</artifactId>
430     <version>11.2</version>
431 </dependency>
```

432
433 3) MySQL의 경우, MySQL Connector/J로 들어가서

```
434 <dependency>
435     <groupId>mysql</groupId>
436     <artifactId>mysql-connector-java</artifactId>
437     <version>8.0.26</version>
438 </dependency>
```

439
440 4) MariaDB의 경우, MariaDB Java Client로 들어가서

```
441 <dependency>
442     <groupId>org.mariadb.jdbc</groupId>
443     <artifactId>mariadb-java-client</artifactId>
444     <version>2.7.4</version>
445 </dependency>
```

446
447 5) pom.xml > right-click > Run As > Maven install
448 [INFO] BUILD SUCCESS 확인
449
450

451 7. Spring JDBC pom.xml에 추가하기

452 1) pom.xml에 다음 코드 추가

```
453 <dependency>
454     <groupId>org.springframework</groupId>
455     <artifactId>spring-jdbc</artifactId>
456     <version>5.3.10</version>
457 </dependency>
```

458
459
460 2) pom.xml > right-click > Run As > Maven install
461 [INFO] BUILD SUCCESS 확인
462
463

464 8. Employee class 생성

465 1) com.example > right-click > New > Class

466 2) Name : Employee

3)Finish

```
package com.example;

import lombok.AllArgsConstructor;
import lombok.Getter;
import lombok.NoArgsConstructor;
import lombok.Setter;

@Getter
@AllArgsConstructor
@NoArgsConstructor
public class Employee {
    @Setter private int id;
    private String name;
    private float salary;
}
```

9. EmployeeDao class 생성

1)com.example > right-click > New > Class

2)Name : EmployeeDao

3)Finish

```
package com.example;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.stereotype.Repository;

@Repository("empDao")
public class EmployeeDao {
    @Autowired
    private JdbcTemplate jdbcTemplate;

    public int saveEmployee(Employee e){
        String query="INSERT INTO Employee
        VALUES('"+e.getId()+"','"+e.getName()+"','"+e.getSalary()+"");
        return jdbcTemplate.update(query);
    }

    public int updateEmployee(Employee e){
        String query="Update Employee SET name='"+e.getName()+"',salary='"+e.getSalary()+"
        where id='"+e.getId()+"' ";
        return jdbcTemplate.update(query);
    }

    public int deleteEmployee(Employee e){
        String query="DELETE FROM Employee where id='"+e.getId()+"' ";
        return jdbcTemplate.update(query);
    }
}
```

10. resources folder 생성하기

1)JdbcTemplateDemo project > right-click > New > Source Folder

2)Folder name : resources

3)Finish

11. resources/dbinfo.properties file 생성

1)Oracle Database 일 경우

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

2)MySQL Database 인 경우

```
db.driverClass=com.mysql.jdbc.Driver
```



```
532 db.url=jdbc:mysql://localhost:3306/world
533 db.username=root
534 db.password=javamysql
535
536
```

537 12. Java Annotation 환경설정 파일 생성

- 538 1)com.example > right-click > New > Class
- 539 2)Name : AppConfig
- 540 3)Finish

```
541
542 package com.example;
543
544 import javax.sql.DataSource;
545
546 import org.springframework.beans.factory.annotation.Value;
547 import org.springframework.context.annotation.Bean;
548 import org.springframework.context.annotation.ComponentScan;
549 import org.springframework.context.support.PropertySourcesPlaceholderConfigurer;
550 import org.springframework.core.io.ClassPathResource;
551 import org.springframework.jdbc.core.JdbcTemplate;
552 import org.springframework.jdbc.datasource.DriverManagerDataSource;
553
554 @ComponentScan(basePackages = "com.example")
555 public class AppConfig {
556     @Value("${db.driverClass}")
557     private String driverClassName;
558     @Value("${db.url}")
559     private String url;
560     @Value("${db.username}")
561     private String username;
562     @Value("${db.password}")
563     private String password;
564
565     @Bean
566     public static PropertySourcesPlaceholderConfigurer properties() {
567         PropertySourcesPlaceholderConfigurer configurator = new
            PropertySourcesPlaceholderConfigurer();
568         configurator.setLocation(new ClassPathResource("dbinfo.properties"));
569         return configurator;
570     }
571
572     @Bean
573     public DataSource dataSource() {
574         DriverManagerDataSource ds = new DriverManagerDataSource();
575         ds.setDriverClassName(this.driverClassName);
576         ds.setUrl(this.url);
577         ds.setUsername(this.username);
578         ds.setPassword(this.password);
579         return ds;
580     }
581
582     @Bean
583     public JdbcTemplate jdbcTemplate() {
584         JdbcTemplate template = new JdbcTemplate();
585         template.setDataSource(this.dataSource());
586         return template;
587     }
588 }
589
590
```

591 13. MainClass Class 생성

- 592 1)com.example > right-click > New > Class
- 593 2)Name : MainClass
- 594 3)Finish

```
595
596 package com.example;
597
```

```

598 import org.springframework.context.ApplicationContext;
599 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
600
601 public class MainClass {
602     public static void main(String[] args) {
603         ApplicationContext ctx=new AnnotationConfigApplicationContext(ApplicationConfig.class);
604
605         EmployeeDao dao = (EmployeeDao)ctx.getBean("empDao");
606         int status = dao.saveEmployee(new Employee(102,"Amit",35000));
607         System.out.println("Insert Status = " + status);
608
609         status = dao.updateEmployee(new Employee(102,"Sonoo",15000));
610         System.out.println("Update Status = " + status);
611
612         Employee e=new Employee();
613         e.setId(102);
614         status = dao.deleteEmployee(e);
615         System.out.println("Delete Status = " + status);
616     }
617 }

```

4)결과

```

620 Insert Status = 1
621 Update Status = 1
622 Delete Status = 1

```

Task3. Example of PreparedStatement in Spring JdbcTemplate

1. EmployeeDao1 class 생성

- 1)com.example.EmployeeDao를 복사하여 붙여넣기
- 2)이름변경 : EmployeeDao1
- 3)OK

```

632 package com.example;
633
634 import java.sql.PreparedStatement;
635 import java.sql.SQLException;
636
637 import org.springframework.beans.factory.annotation.Autowired;
638 import org.springframework.dao.DataAccessException;
639 import org.springframework.jdbc.core.JdbcTemplate;
640 import org.springframework.jdbc.core.PreparedStatementCallback;
641 import org.springframework.stereotype.Repository;
642
643 @Repository("empDao1")
644 public class EmployeeDao1 {
645     @Autowired
646     private JdbcTemplate jdbcTemplate;
647
648     public Boolean saveEmployeeByPreparedStatement(final Employee e){
649         String query="INSERT INTO Employee VALUES(?,?,?)";
650         return jdbcTemplate.execute(query,new PreparedStatementCallback<Boolean>(){
651             @Override
652             public Boolean doInPreparedStatement(PreparedStatement ps)
653                 throws SQLException, DataAccessException {
654
655                 ps.setInt(1,e.getId());
656                 ps.setString(2,e.getName());
657                 ps.setFloat(3,e.getSalary());
658
659                 return ps.execute();
660             }
661         });
662     }
663 }

```

```

665
666
667 2. ApplicationConfig1 class 생성
668 1)com.example.ApplicationConfig.java를 복사하여 붙여넣기
669 2)이름변경 : ApplicationConfig1
670 3)OK
671
672 package com.example;
673
674 import javax.sql.DataSource;
675
676 import org.springframework.beans.factory.annotation.Value;
677 import org.springframework.context.annotation.Bean;
678 import org.springframework.context.annotation.ComponentScan;
679 import org.springframework.context.annotation.Configuration;
680 import org.springframework.context.support.PropertySourcesPlaceholderConfigurer;
681 import org.springframework.core.io.ClassPathResource;
682 import org.springframework.jdbc.core.JdbcTemplate;
683 import org.springframework.jdbc.datasource.DriverManagerDataSource;
684
685 @Configuration
686 @ComponentScan(basePackages = "com.example")
687 public class ApplicationConfig1 {
688     @Value("${db.driverClass}")
689     private String driverClassName;
690     @Value("${db.url}")
691     private String url;
692     @Value("${db.username}")
693     private String username;
694     @Value("${db.password}")
695     private String password;
696
697     @Bean
698     public static PropertySourcesPlaceholderConfigurer properties() {
699         PropertySourcesPlaceholderConfigurer configurer = new
        PropertySourcesPlaceholderConfigurer();
700         configurer.setLocation(new ClassPathResource("dbinfo.properties"));
701         return configurer;
702     }
703
704     @Bean
705     public DataSource dataSource() {
706         DriverManagerDataSource ds = new DriverManagerDataSource();
707         ds.setDriverClassName(this.driverClassName);
708         ds.setUrl(this.url);
709         ds.setUsername(this.username);
710         ds.setPassword(this.password);
711         return ds;
712     }
713
714     @Bean
715     public JdbcTemplate jdbcTemplate() {
716         JdbcTemplate template = new JdbcTemplate();
717         template.setDataSource(this.dataSource());
718         return template;
719     }
720
721     @Bean
722     public EmployeeDao1 empDao1() {
723         EmployeeDao1 empDao1 = new EmployeeDao1();
724         return empDao1;
725     }
726 }
727
728 4)applicationContext.xml를 사용할 경우
729 <?xml version="1.0" encoding="UTF-8"?>
730 <beans

```

```

731 xmlns="http://www.springframework.org/schema/beans"
732 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
733 xmlns:p="http://www.springframework.org/schema/p"
734 xsi:schemaLocation="http://www.springframework.org/schema/beans
735 http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
736
737 <bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
738     <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
739     <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />
740     <property name="username" value="scott" />
741     <property name="password" value="tiger" />
742 </bean>
743
744 <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
745     <property name="dataSource" ref="ds" />
746 </bean>
747
748 <bean id="empDao1" class="com.example.EmployeeDao1">
749     <property name="jdbcTemplate" ref="jdbcTemplate" />
750 </bean>
751 </beans>

```

754 3. MainClass1 class 생성

- 755 1)com.example.MainClass 복사하여 붙여넣기
- 756 2)이름변경 : MainClass1
- 757 3)OK

```
759 package com.example;
```

```
761 import org.springframework.context.ApplicationContext;
```

```
762 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
```

```
764 public class Test {
```

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

```
766         ApplicationContext ctx=new AnnotationConfigApplicationContext(ApplicationConfig1.class);
```

```
768         EmployeeDao1 dao1 = (EmployeeDao1)ctx.getBean("empDao1");
```

```
769         dao1.saveEmployeeByPreparedStatement(new Employee(108,"Amit",35000));
```

```
770     }
```

```
771 }
```

773 4)applicationContext.xml을 사용할 경우 Test.java

```
774 package com.example;
```

```
776 import org.springframework.context.ApplicationContext;
```

```
777 import org.springframework.context.support.ClassPathXmlApplicationContext;
```

```
778 public class Test {
```

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

```
781         ApplicationContext ctx=new ClassPathXmlApplicationContext("applicationContext.xml");
```

```
783         EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");
```

```
784         dao.saveEmployeeByPreparedStatement(new Employee(108,"Amit",35000));
```

```
785     }
```

```
786 }
```

791 Task4. ResultSetExtractor Example | Fetching Records by Spring JdbcTemplate

792 1. Employee.java 수정

```
794 package com.example;
```

```
796 import lombok.AllArgsConstructor;
```

```
797 import lombok.Getter;
```

```

798 import lombok.NoArgsConstructor;
799 import lombok.Setter;
800 import lombok.ToString;
801
802 @Getter
803 @Setter
804 @AllArgsConstructor
805 @NoArgsConstructor
806 @ToString
807 public class Employee {
808     private int id;
809     private String name;
810     private float salary;
811 }

```

```

812
813
814 2. EmployeeDao2 class 생성
815 1)com.example.EmployeeDao1를 복사하여 붙여넣기
816 2)이름변경 : EmployeeDao2
817 3)OK
818
819 package com.example;
820
821 import java.sql.ResultSet;
822 import java.sql.SQLException;
823 import java.util.ArrayList;
824 import java.util.List;
825
826 import org.springframework.beans.factory.annotation.Autowired;
827 import org.springframework.dao.DataAccessException;
828 import org.springframework.jdbc.core.JdbcTemplate;
829 import org.springframework.jdbc.core.ResultSetExtractor;
830 import org.springframework.stereotype.Repository;
831
832 @Repository("empDao2")
833 public class EmployeeDao2 {
834     @Autowired
835     private JdbcTemplate jdbcTemplate;
836
837     public List<Employee> getAllEmployees(){
838         return jdbcTemplate.query("SELECT * FROM Employee", new
            ResultSetExtractor<List<Employee>>(){
839             @Override
840             public List<Employee> extractData(ResultSet rs) throws SQLException,
841                 DataAccessException {
842
843                 List<Employee> list=new ArrayList<Employee>();
844                 while(rs.next()){
845                     Employee e=new Employee();
846                     e.setId(rs.getInt(1));
847                     e.setName(rs.getString(2));
848                     e.setSalary(rs.getFloat(3));
849                     list.add(e);
850                 }
851                 return list;
852             }
853         });
854     }
855 }

```

```

856
857
858 3. ApplicationConfig2 class 생성
859 1)com.example.ApplicationConfig1.java를 복사하여 붙여넣기
860 2)이름변경 : ApplicationConfig2
861 3)OK
862
863 ...

```

```

864     @Bean
865     public EmployeeDao2 empDao2() {
866         EmployeeDao2 empDao2 = new EmployeeDao2();
867         return empDao2;
868     }
869 }

```

4)applicationContext.xml를 사용할 경우

```

872 <?xml version="1.0" encoding="UTF-8"?>
873 <beans
874     xmlns="http://www.springframework.org/schema/beans"
875     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
876     xmlns:p="http://www.springframework.org/schema/p"
877     xsi:schemaLocation="http://www.springframework.org/schema/beans
878         http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
879
880     <bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
881         <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
882         <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />
883         <property name="username" value="scott" />
884         <property name="password" value="tiger" />
885     </bean>
886
887     <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
888         <property name="dataSource" ref="ds" />
889     </bean>
890
891     <bean id="empDao2" class="com.example.EmployeeDao2">
892         <property name="jdbcTemplate" ref="jdbcTemplate" />
893     </bean>
894
895 </beans>

```

4. MainClass2 class 생성

- 899 1)MainClass1.java copy하여 붙여기
- 900 2)이름변경 : MainClass2
- 901 3)OK

```

902
903     package com.example;
904
905     import org.springframework.context.ApplicationContext;
906     import org.springframework.context.annotation.AnnotationConfigApplicationContext;
907
908     public class MainClass2 {
909         public static void main(String[] args) {
910             ApplicationContext ctx=new AnnotationConfigApplicationContext(ApplicationConfig2.class);
911
912             EmployeeDao2 dao2 = (EmployeeDao2)ctx.getBean("empDao2");
913             dao2.getAllEmployees().forEach(emp -> System.out.println(emp));
914         }
915     }
916

```

4)결과

```

918     Employee(id=108, name=Amit, salary=35000.0)
919

```

5)applicationContext.xml을 사용할 경우 Test.java

```

921     package com.example;
922
923     import java.util.List;
924
925     import org.springframework.context.ApplicationContext;
926     import org.springframework.context.support.ClassPathXmlApplicationContext;
927     public class Test {
928
929         public static void main(String[] args) {
930             ApplicationContext ctx=new ClassPathXmlApplicationContext("applicationContext.xml");

```

```

931         EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");
932         List<Employee> list=dao.getAllEmployees();
933
934         for(Employee e:list)
935             System.out.println(e);
936
937     }
938 }
939
940
941
942 -----

```

943 Task5. RowMapper Example | Fetching records by Spring JdbcTemplate

944 1. EmployeeDao3 class 생성

- 945 1)com.example.EmployeeDao2를 복사하여 붙여넣기
- 946 2)이름변경 : EmployeeDao3
- 947 3)OK

```

949     package com.example;
950
951     import java.sql.ResultSet;
952     import java.sql.SQLException;
953     import java.util.List;
954
955     import org.springframework.beans.factory.annotation.Autowired;
956     import org.springframework.jdbc.core.JdbcTemplate;
957     import org.springframework.jdbc.core.RowMapper;
958     import org.springframework.stereotype.Repository;
959
960     @Repository("empDao3")
961     public class EmployeeDao3 {
962         @Autowired
963         private JdbcTemplate jdbcTemplate;
964
965         public List<Employee> getAllEmployeesRowMapper(){
966             return jdbcTemplate.query("select * from employee",new RowMapper<Employee>(){
967                 @Override
968                 public Employee mapRow(ResultSet rs, int rownumber) throws SQLException {
969                     Employee e=new Employee();
970                     e.setId(rs.getInt(1));
971                     e.setName(rs.getString(2));
972                     e.setSalary(rs.getFloat(3));
973                     return e;
974                 }
975             });
976         }
977     }
978
979

```

980 2. ApplicationConfig3 class 생성

- 981 1)com.example.ApplicationConfig2.java를 복사하여 붙여넣기
- 982 2)이름변경 : ApplicationConfig3
- 983 3)OK

```

985     ...
986     @Bean
987     public EmployeeDao3 empDao3() {
988         EmployeeDao3 empDao3 = new EmployeeDao3();
989         return empDao3;
990     }
991

```

992 4)applicationContext.xml 를 사용할 경우

```

993     <?xml version="1.0" encoding="UTF-8"?>
994     <beans
995         xmlns="http://www.springframework.org/schema/beans"
996         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
997         xmlns:p="http://www.springframework.org/schema/p"

```

```

998      xsi:schemaLocation="http://www.springframework.org/schema/beans
999      http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">
1000
1001      <bean id="ds" class="org.springframework.jdbc.datasource.DriverManagerDataSource">
1002          <property name="driverClassName" value="oracle.jdbc.driver.OracleDriver" />
1003          <property name="url" value="jdbc:oracle:thin:@localhost:1521:xe" />
1004          <property name="username" value="scott" />
1005          <property name="password" value="tiger" />
1006      </bean>
1007
1008      <bean id="jdbcTemplate" class="org.springframework.jdbc.core.JdbcTemplate">
1009          <property name="dataSource" ref="ds" />
1010      </bean>
1011
1012      <bean id="empDao3" class="com.example.EmployeeDao3">
1013          <property name="jdbcTemplate" ref="jdbcTemplate"></property>
1014      </bean>
1015
1016  </beans>
1017
1018
1019  3. MainClass3 class 생성
1020      1)MainClass2.java copy하여 붙여기
1021      2)이름변경 : MainClass3
1022      3)OK
1023
1024      package com.example;
1025
1026      import org.springframework.context.ApplicationContext;
1027      import org.springframework.context.annotation.AnnotationConfigApplicationContext;
1028
1029      public class MainClass3 {
1030          public static void main(String[] args) {
1031              ApplicationContext ctx=new AnnotationConfigApplicationContext(ApplicationConfig3.class);
1032
1033              EmployeeDao3 dao3 = (EmployeeDao3)ctx.getBean("empDao3");
1034              dao3.getAllEmployeesRowMapper().forEach(emp -> System.out.println(emp));
1035          }
1036      }
1037
1038  4)결과
1039      Employee(id=108, name=Amit, salary=35000.0)
1040
1041  5)applicationContext.xml을 사용할 경우 Test.java
1042      package com.example;
1043
1044      import java.util.List;
1045
1046      import org.springframework.context.ApplicationContext;
1047      import org.springframework.context.support.ClassPathXmlApplicationContext;
1048      public class Test {
1049
1050          public static void main(String[] args) {
1051              ApplicationContext ctx=new ClassPathXmlApplicationContext("applicationContext.xml");
1052              EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");
1053              List<Employee> list=dao.getAllEmployeesRowMapper();
1054
1055              for(Employee e:list)
1056                  System.out.println(e);
1057          }
1058      }
1059
1060
1061
1062
1063 -----
1064 Task6. Spring NamedParameterJdbcTemplate Example

```



```
1065 1. Create Table
1066
1067 CREATE TABLE Employee(
1068     id NUMBER(10),
1069     name VARCHAR2(100),
1070     salary NUMBER(10)
1071 );
1072
1073
1074 2. In Package Explorer > right-click > New > Java Project
1075     1)Project Name : NamedJdbcTemplateDemo
1076     2)JRE
1077         -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
1078     3)Uncheck [Create module-info.java file]
1079     4)Next
1080     5)Finish
1081
1082
1083 3. src > right-click > New > Package
1084     1)Name : com.example
1085     2)Finish
1086
1087
1088 4. Java Project를 Spring Project로 변환
1089     1)NamedJdbcTemplateDemo Project > right-click > Configure > Convert to Maven Project
1090         -Project : /NamedJdbcTemplateDemo
1091         -Group Id : NamedJdbcTemplateDemo
1092         -Artifact Id : NamedJdbcTemplateDemo
1093         -version : 0.0.1-SNAPSHOT
1094         -Packaging : jar
1095         -Finish
1096
1097     2)NamedJdbcTemplateDemo Project > right-click > Spring > Add Spring Project Nature
1098
1099     3)pom.xml file에 Spring Context Dependency 추가하기
1100         <version>0.0.1-SNAPSHOT</version>
1101         <dependencies>
1102             <dependency>
1103                 <groupId>org.springframework</groupId>
1104                 <artifactId>spring-context</artifactId>
1105                 <version>5.3.10</version>
1106             </dependency>
1107         </dependencies>
1108
1109     4)pom.xml > right-click > Run As > Maven install
1110         [INFO] BUILD SUCCESS 확인
1111
1112
1113 5. Lombok library 추가
1114     1)https://mvnrepository.com/에서 'lombok'으로 검색
1115     2)'Project Lombok' click
1116     3)1.18.20 click
1117     4)dependency copy해서 pom.xml에 붙여넣기
1118
1119         <dependency>
1120             <groupId>org.projectlombok</groupId>
1121             <artifactId>lombok</artifactId>
1122             <version>1.18.20</version>
1123             <scope>provided</scope>
1124         </dependency>
1125
1126     5)pom.xml > right-click > Run As > Maven install
1127         [INFO] BUILD SUCCESS 확인
1128
1129
1130 6. pom.xml에 Jdbc Driver 설정하기
1131     1)Oracle 12C 이후 version일 경우, mvnrepository에서 oralc로 검색후, Ojdbc8 설치
```

```
1132 <dependency>
1133 <groupId>com.oracle.database.jdbc</groupId>
1134 <artifactId>ojdbc8</artifactId>
1135 <version>21.3.0.0</version>
1136 </dependency>
```

1137 2)Oracle 11g version일 경우

1138 -pom.xml에 붙여 넣고 Maven Install 하기

```
1139 <dependency>
1140 <groupId>com.oracle</groupId>
1141 <artifactId>ojdbc6</artifactId>
1142 <version>11.2</version>
1143 </dependency>
```

1144 3)MySQL의 경우, MySQL Connector/J로 들어가서

```
1145 <dependency>
1146 <groupId>mysql</groupId>
1147 <artifactId>mysql-connector-java</artifactId>
1148 <version>8.0.26</version>
1149 </dependency>
```

1150 4)MariaDB의 경우, MariaDB Java Client로 들어가서

```
1151 <dependency>
1152 <groupId>org.mariadb.jdbc</groupId>
1153 <artifactId>mariadb-java-client</artifactId>
1154 <version>2.7.4</version>
1155 </dependency>
```

1156 5)pom.xml > right-click > Run As > Maven install

1157 [INFO] BUILD SUCCESS 확인

1162 7. Spring JDBC pom.xml에 추가하기

1163 1)pom.xml에 다음 코드 추가

```
1164 <dependency>
1165 <groupId>org.springframework</groupId>
1166 <artifactId>spring-jdbc</artifactId>
1167 <version>5.3.10</version>
1168 </dependency>
```

1169 2)pom.xml > right-click > Run As > Maven install

1170 [INFO] BUILD SUCCESS 확인

1176 8. Employee class 생성

1177 1)com.example > right-click > New > Class

1178 2)Name : Employee

1179 3)Finish

```
1180 package com.example;
1181
1182 import lombok.AllArgsConstructor;
1183 import lombok.Getter;
1184
1185 @Getter
1186 @AllArgsConstructor
1187 public class Employee {
1188     private int id;
1189     private String name;
1190     private float salary;
1191 }
1192
```

1193 9. EmployeeDao class 생성

1194 1)com.example > New > Class

1195 2)Name : EmployeeDao

3)Finish

```
package com.example;

import java.sql.PreparedStatement;
import java.sql.SQLException;
import java.util.HashMap;
import java.util.Map;

import org.springframework.beans.factory.annotation.Autowired;
import org.springframework.dao.DataAccessException;
import org.springframework.jdbc.core.PreparedStatementCallback;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.stereotype.Repository;

@Repository
public class EmployeeDao {
    @Autowired
    private NamedParameterJdbcTemplate template;

    public void save (Employee emp){
        String query="INSERT INTO Employee VALUES (:id,:name,:salary)";

        Map<String,Object> map = new HashMap<String,Object>();
        map.put("id", emp.getId());
        map.put("name", emp.getName());
        map.put("salary", emp.getSalary());

        template.execute(query, map, new PreparedStatementCallback<Integer>() {
            @Override
            public Integer doInPreparedStatement(PreparedStatement ps)
                throws SQLException, DataAccessException {
                return ps.executeUpdate();
            }
        });
    }
}
```

10. resources folder 생성하기

- 1)NamedJdbcTemplateDemo project > right-click > New > Source Folder
- 2)Folder name : resources
- 3)Finish

11. resources/dbinfo.properties file 생성

- 1)Oracle Database 인 경우
db.driverClass=oracle.jdbc.driver.OracleDriver
db.url=jdbc:oracle:thin:@localhost:1521:XE
db.username=hr
db.password=hr
- 2)MySQL 인 경우
db.driverClass=com.mysql.jdbc.Driver
db.url=jdbc:mysql://localhost:3306/world
db.username=root
db.password=javamysql

12. ApplicationConfig class 생성

- 1)com.example > New > Class
- 2)Name : ApplicationConfig
- 3)Finish

```
package com.example;

import javax.sql.DataSource;
```

```

import org.springframework.beans.factory.annotation.Value;
import org.springframework.context.annotation.Bean;
import org.springframework.context.annotation.ComponentScan;
import org.springframework.context.annotation.Configuration;
import org.springframework.context.support.PropertySourcesPlaceholderConfigurer;
import org.springframework.core.io.ClassPathResource;
import org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate;
import org.springframework.jdbc.datasource.DriverManagerDataSource;

```

```

@Configuration
@ComponentScan(basePackages = "com.example")
public class ApplicationConfig {
    @Value("${db.driverClass}")
    private String driverClassName;
    @Value("${db.url}")
    private String url;
    @Value("${db.username}")
    private String username;
    @Value("${db.password}")
    private String password;

    @Bean
    public static PropertySourcesPlaceholderConfigurer properties() {
        PropertySourcesPlaceholderConfigurer configurator = new
            PropertySourcesPlaceholderConfigurer();
        configurator.setLocation(new ClassPathResource("dbinfo.properties"));
        return configurator;
    }

    @Bean
    public DataSource dataSource() {
        DriverManagerDataSource ds = new DriverManagerDataSource();
        ds.setDriverClassName(this.driverClassName);
        ds.setUrl(this.url);
        ds.setUsername(this.username);
        ds.setPassword(this.password);
        return ds;
    }

    @Bean
    public NamedParameterJdbcTemplate jdbcTemplate() {
        NamedParameterJdbcTemplate template = new
            NamedParameterJdbcTemplate(this.dataSource());
        return template;
    }

    @Bean
    public EmployeeDao empDao() {
        EmployeeDao empDao = new EmployeeDao();
        return empDao;
    }
}

```

4)applicationContext.xml를 사용할 경우

```

<?xml version="1.0" encoding="UTF-8"?>
<beans
    xmlns="http://www.springframework.org/schema/beans"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xmlns:p="http://www.springframework.org/schema/p"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-3.0.xsd">

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

```

1331         <property name="password" value="tiger" />
1332     </bean>
1333
1334     <bean id="jtemplate"
1335           class="org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate">
1336         <constructor-arg ref="ds" />
1337     </bean>
1338
1339     <bean id="empDao" class="com.example.EmployeeDao">
1340         <constructor-arg>
1341             <ref bean="jtemplate"/>
1342         </constructor-arg>
1343     </bean>
1344 </beans>

```

13. MainClass class 생성

- 1)com.example > New > Class
- 2)Name : MainClass
- 3)Finish

```

1352 package com.example;
1353
1354 import org.springframework.context.ApplicationContext;
1355 import org.springframework.context.annotation.AnnotationConfigApplicationContext;
1356
1357 public class MainClass {
1358     public static void main(String[] args) {
1359         ApplicationContext ctx = new AnnotationConfigApplicationContext(ApplicationConfig.class);
1360
1361         EmployeeDao dao=(EmployeeDao)ctx.getBean("empDao");
1362         dao.save(new Employee(23,"sonoo",50000));
1363     }
1364 }

```

4)applicationContext.xml을 사용할 경우 Test.java

```

1368 package com.example;
1369
1370 import org.springframework.beans.factory.BeanFactory;
1371 import org.springframework.beans.factory.xml.XmlBeanFactory;
1372 import org.springframework.core.io.ClassPathResource;
1373 import org.springframework.core.io.Resource;
1374
1375 public class SimpleTest {
1376     public static void main(String[] args) {
1377
1378         Resource r=new ClassPathResource("applicationContext.xml");
1379         BeanFactory factory=new XmlBeanFactory(r);
1380
1381         EmpDao dao=(EmpDao)factory.getBean("edao");
1382         dao.save(new Emp(23,"sonoo",50000));
1383     }
1384 }

```

Task7. Calling Stored Procedure

1. Preparation Table & Stored Procedure

1)Create Table

```

1394 CREATE TABLE Student(
1395     id INT NOT NULL AUTO_INCREMENT,
1396     name VARCHAR(20) NOT NULL,

```

```

1397         age INT NOT NULL,
1398         PRIMARY KEY(id)
1399     );
1400
1401 2)Create Stored Procedure
1402 DROP PROCEDURE IF EXISTS sp_student_select;
1403 delimiter //
1404 CREATE PROCEDURE sp_student_select
1405 (
1406     IN    v_id INT,
1407     OUT   v_name VARCHAR(20),
1408     OUT   v_age INT
1409 )
1410 BEGIN
1411     SELECT name, age INTO v_name, v_age
1412     FROM Student
1413     WHERE id = v_id;
1414 END; //
1415 delimiter ;
1416
1417 DROP PROCEDURE IF EXISTS sp_student_selectlist;
1418 delimiter //
1419 CREATE PROCEDURE sp_student_selectlist()
1420 BEGIN
1421     SELECT *
1422     FROM Student;
1423 END; //
1424 delimiter ;
1425
1426 DROP PROCEDURE IF EXISTS sp_student_insert;
1427 delimiter //
1428 CREATE PROCEDURE sp_student_insert
1429 (
1430     IN    v_name VARCHAR(20),
1431     IN    v_age INT
1432 )
1433 BEGIN
1434     INSERT INTO Student(name, age)
1435     VALUES(v_name, v_age);
1436     COMMIT;
1437 END; //
1438 delimiter ;
1439
1440 DROP PROCEDURE IF EXISTS sp_student_update;
1441 delimiter //
1442 CREATE PROCEDURE sp_student_update
1443 (
1444     IN    v_id INT,
1445     IN    v_age INT
1446 )
1447 BEGIN
1448     UPDATE Student SET age = v_age
1449     WHERE id = v_id;
1450     COMMIT;
1451 END; //
1452 delimiter ;
1453
1454 DROP PROCEDURE IF EXISTS sp_student_delete;
1455 delimiter //
1456 CREATE PROCEDURE sp_student_delete
1457 (
1458     IN    v_id INT
1459 )
1460 BEGIN
1461     DELETE FROM Student
1462     WHERE id = v_id;
1463     COMMIT;

```

```
1464     END; //
1465     delimiter ;
1466
1467
1468 2. In Package Explorer > right-click > New > Java Project
1469     1)Project Name : JdbcTemplateDemo2
1470     2)JRE
1471         -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
1472     3)Uncheck [Create module-info.java file]
1473     4)Next
1474     5)Finish
1475
1476
1477 3. src > right-click > New > Package
1478     1)Name : com.example
1479     2)Finish
1480
1481
1482 4. Java Project를 Spring Project로 변환
1483     1)JdbcTemplateDemo2 Project > right-click > Configure > Convert to Maven Project
1484         -Project : /JdbcTemplateDemo2
1485         -Group Id : JdbcTemplateDemo2
1486         -Artifact Id : JdbcTemplateDemo2
1487         -version : 0.0.1-SNAPSHOT
1488         -Packaging : jar
1489         -Finish
1490
1491     2)JdbcTemplateDemo2 Project > right-click > Spring > Add Spring Project Nature
1492
1493     3)pom.xml file에 Spring Context Dependency 추가하기
1494         <version>0.0.1-SNAPSHOT</version>
1495         <dependencies>
1496             <dependency>
1497                 <groupId>org.springframework</groupId>
1498                 <artifactId>spring-context</artifactId>
1499                 <version>5.3.10</version>
1500             </dependency>
1501         </dependencies>
1502
1503     4)pom.xml > right-click > Run As > Maven install
1504         [INFO] BUILD SUCCESS 확인
1505
1506
1507 5. Lombok library 추가
1508     1)https://mvnrepository.com/에서 'lombok'으로 검색
1509     2)'Project Lombok' click
1510     3)1.18.20 click
1511     4)dependency copy해서 pom.xml에 붙여넣기
1512
1513         <dependencies>
1514             <dependency>
1515                 <groupId>org.springframework</groupId>
1516                 <artifactId>spring-context</artifactId>
1517                 <version>5.3.10</version>
1518             </dependency>
1519             <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
1520             <dependency>
1521                 <groupId>org.projectlombok</groupId>
1522                 <artifactId>lombok</artifactId>
1523                 <version>1.18.20</version>
1524                 <scope>provided</scope>
1525             </dependency>
1526         </dependencies>
1527
1528     5)pom.xml > right-click > Run As > Maven install
1529         [INFO] BUILD SUCCESS 확인
1530
```

1531
1532 6. pom.xml에 Jdbc Driver 설정하기
1533 1)Oracle 12C 이후 version일 경우, mvnrepository에서 oralcce로 검색후, Ojdbc8 설치
1534 <dependency>
1535 <groupId>com.oracle.database.jdbc</groupId>
1536 <artifactId>ojdbc8</artifactId>
1537 <version>21.3.0.0</version>
1538 </dependency>
1539
1540 2)Oracle 11g version일 경우
1541 -pom.xml에 붙여 넣고 Maven Install 하기
1542 <dependency>
1543 <groupId>com.oracle</groupId>
1544 <artifactId>ojdbc6</artifactId>
1545 <version>11.2</version>
1546 </dependency>
1547
1548 3)MySQL의 경우, MySQL Connector/J로 들어가서
1549 <dependency>
1550 <groupId>mysql</groupId>
1551 <artifactId>mysql-connector-java</artifactId>
1552 <version>8.0.26</version>
1553 </dependency>
1554
1555 4)MariaDB의 경우, MariaDB Java Client로 들어가서
1556 <dependency>
1557 <groupId>org.mariadb.jdbc</groupId>
1558 <artifactId>mariadb-java-client</artifactId>
1559 <version>2.7.4</version>
1560 </dependency>
1561
1562 5)pom.xml > right-click > Run As > Maven install
1563 [INFO] BUILD SUCCESS 확인
1564
1565
1566 7. Spring JDBC pom.xml에 추가하기
1567 1)pom.xml에 다음 코드 추가
1568
1569 <dependency>
1570 <groupId>org.springframework</groupId>
1571 <artifactId>spring-jdbc</artifactId>
1572 <version>5.3.10</version>
1573 </dependency>
1574
1575 2)pom.xml > right-click > Run As > Maven install
1576 [INFO] BUILD SUCCESS 확인
1577
1578
1579 8. Student class 생성
1580 1)com.example > right-click > New > Class
1581 2)Name : Student
1582 3)Finish
1583
1584 package com.example;
1585
1586 import lombok.AllArgsConstructor;
1587 import lombok.Getter;
1588 import lombok.NoArgsConstructor;
1589 import lombok.Setter;
1590
1591 @Getter
1592 @Setter
1593 @AllArgsConstructor
1594 @NoArgsConstructor
1595 public class Student {
1596 private int id;
1597 private String name;


```

1598         private int age;
1599     }
1600
1601
1602 9. StudentDao interface 생성
1603     1)com.example > right-click > New > Interface
1604     2)Name : StudentDao
1605     3)Finish
1606
1607         package com.example;
1608
1609         import java.util.List;
1610         import javax.sql.DataSource;
1611
1612         public class StudentDao {
1613             void setDataSource(DataSource ds);
1614             void create(String name, int age);
1615             void delete(int id);
1616             void updae(int id, int age);
1617             Student getStudent(int id);
1618             List<Student> listStudents();
1619         }
1620
1621
1622 10. resources folder 생성하기
1623     1)JdbcTemplateDemo project > right-click > New > Source Folder
1624     2)Folder name : resources
1625     3)Finish
1626
1627
1628 11. resources/dbinfo.properties file 생성
1629     1)Oracle Database 일 경우
1630         db.driverClass=oracle.jdbc.driver.OracleDriver
1631         db.url=jdbc:oracle:thin:@localhost:1521:XE
1632         db.username=hr
1633         db.password=hr
1634
1635     2)MySQL Database 인 경우
1636         db.driverClass=com.mysql.jdbc.Driver
1637         db.url=jdbc:mysql://localhost:3306/world
1638         db.username=root
1639         db.password=javamysql
1640
1641
1642 12. src/com.example.StudentMapper class 생성
1643     1)com.example > right-click > New > Class
1644     2)Name : StudentMapper
1645     3)Finish
1646
1647         package com.example;
1648
1649         import java.sql.ResultSet;
1650         import java.sql.SQLException;
1651
1652         import org.springframework.jdbc.core.RowMapper;
1653
1654         public class StudentMapper implements RowMapper<Student>{
1655             public Student mapRow(ResultSet rs, int rowNum) throws SQLException{
1656                 Student student = new Student();
1657                 student.setId(rs.getInt("id"));
1658                 student.setName(rs.getString("name"));
1659                 student.setAge(rs.getInt("age"));
1660                 return student;
1661             }
1662         }
1663
1664

```

```

1665 13. StudentJDBCTemplate class 생성 ==> SimpleJdbcCall 사용하기
1666     1)com.example > right-click > New > Class
1667     2)Name : StudentJDBCTemplate
1668     3)Finish
1669
1670     package com.example;
1671
1672     import java.sql.Types;
1673     import java.util.ArrayList;
1674     import java.util.List;
1675     import java.util.Map;
1676     import java.util.Set;
1677
1678     import javax.sql.DataSource;
1679
1680     import org.springframework.jdbc.core.JdbcTemplate;
1681     import org.springframework.jdbc.core.SqlOutParameter;
1682     import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
1683     import org.springframework.jdbc.core.namedparam.SqlParameterSource;
1684     import org.springframework.jdbc.core.simple.SimpleJdbcCall;
1685
1686     public class StudentJDBCTemplate implements StudentDao {
1687         private DataSource dataSource;
1688         private JdbcTemplate jdbcTemplate;
1689
1690         @Override
1691         public void setDataSource(DataSource ds) {
1692             this.dataSource = ds;
1693             this.jdbcTemplate = new JdbcTemplate(this.dataSource);
1694         }
1695
1696         @Override
1697         public void create(String name, int age) {
1698             SimpleJdbcCall jdbcCall =
1699                 new SimpleJdbcCall(this.dataSource).withProcedureName("sp_student_insert");
1700             SqlParameterSource in = new MapSqlParameterSource().addValue("v_name", name)
1701                 .addValue("v_age", age);
1702             jdbcCall.execute(in);
1703         }
1704
1705         @Override
1706         public void delete(int id) {
1707             SimpleJdbcCall jdbcCall =
1708                 new SimpleJdbcCall(this.dataSource).withProcedureName("sp_student_delete");
1709             SqlParameterSource in = new MapSqlParameterSource().addValue("v_id", id);
1710             jdbcCall.execute(in);
1711         }
1712
1713         @Override
1714         public void update(int id, int age) {
1715             SimpleJdbcCall jdbcCall =
1716                 new SimpleJdbcCall(this.dataSource).withProcedureName("sp_student_update");
1717             SqlParameterSource in = new MapSqlParameterSource().addValue("v_id", id)
1718                 .addValue("v_age", age);
1719             jdbcCall.execute(in);
1720         }
1721
1722         @Override
1723         public Student getStudent(int id) {
1724             SimpleJdbcCall jdbcCall =
1725                 new SimpleJdbcCall(this.dataSource).withProcedureName("sp_student_select")
1726                 .declareParameters(new SqlOutParameter("v_name", Types.VARCHAR),
1727                     new SqlOutParameter("v_age", Types.INTEGER));
1728             SqlParameterSource in = new MapSqlParameterSource().addValue("v_id", id);
1729             Map<String, Object> map = jdbcCall.execute(in);
1730
1731             Student student = new Student();

```

```

1732         student.setId(id);
1733         student.setName((String)map.get("v_name"));
1734         student.setAge((Integer)map.get("v_age"));
1735         //student.setAge(Integer.parseInt(map.get("v_age").toString()));
1736         //Set<String> set = map.keySet();
1737         //System.out.println(set);    //[#update-count-1, v_name, v_age]
1738         return student;
1739     }
1740
1741     @Override
1742     public List<Student> listStudents() {
1743         SimpleJdbcCall jdbcCall =
1744             new SimpleJdbcCall(this.dataSource).withProcedureName("sp_student_selectlist");
1745         SqlParameterSource in = new MapSqlParameterSource();
1746         Map<String, Object> map = jdbcCall.execute(in);
1747         return mapStudents(map, 1);
1748     }
1749
1750     private List<Student> mapStudents(Map<String, Object> out, int resultSetPosition){
1751         List<Student> students = new ArrayList<Student>();
1752         List<Map<String, Object>> results = (List<Map<String, Object>>)out.get("#result-set-1");
1753         results.forEach(s -> {
1754             Student student = new Student();
1755             student.setId((Integer)s.get("id"));
1756             student.setName((String)s.get("name"));
1757             student.setAge((Integer)s.get("age"));
1758             students.add(student);
1759         });
1760         return students;
1761     }
1762 }
1763 }
1764
1765

```

14. StudentJDBCTemplate class 생성 ==> CallableStatementCreator 사용하기

- 1)com.example > right-click > New > Class
- 2)Name : StudentJDBCTemplate1
- 3)Finish

```

1771     package com.example;
1772
1773     import java.sql.CallableStatement;
1774     import java.sql.Connection;
1775     import java.sql.SQLException;
1776     import java.sql.Types;
1777     import java.util.ArrayList;
1778     import java.util.Arrays;
1779     import java.util.List;
1780     import java.util.Map;
1781
1782     import javax.sql.DataSource;
1783
1784     import org.springframework.jdbc.core.CallableStatementCreator;
1785     import org.springframework.jdbc.core.JdbcTemplate;
1786     import org.springframework.jdbc.core.SqlOutParameter;
1787     import org.springframework.jdbc.core.SqlParameter;
1788
1789     public class StudentJDBCTemplate implements StudentDao {
1790         private DataSource dataSource;
1791         private JdbcTemplate jdbcTemplate;
1792
1793         @Override
1794         public void setDataSource(DataSource ds) {
1795             this.dataSource = ds;
1796             this.jdbcTemplate = new JdbcTemplate(this.dataSource);
1797         }
1798     }

```

```

1799 @Override
1800 public void create(String name, int age) {
1801     List<SqlParameter> parameters = Arrays.asList(
1802         new SqlParameter(Types.VARCHAR), new SqlParameter(Types.INTEGER));
1803     this.jdbcTemplate.call(new CallableStatementCreator() {
1804         @Override
1805         public CallableStatement createCallableStatement(Connection conn) throws
1806             SQLException {
1807             CallableStatement cstmt = conn.prepareCall("{call sp_student_insert(?, ?)}");
1808             cstmt.setString(1, name);
1809             cstmt.setInt(2, age);
1810             return cstmt;
1811         }}, parameters);
1812 }
1813
1814 @Override
1815 public void delete(int id) {
1816     List<SqlParameter> parameter = Arrays.asList(new SqlParameter(Types.INTEGER));
1817     this.jdbcTemplate.call(new CallableStatementCreator() {
1818         @Override
1819         public CallableStatement createCallableStatement(Connection con) throws
1820             SQLException {
1821             CallableStatement cstmt = con.prepareCall("{call sp_student_delete(?)}}");
1822             cstmt.setInt(1, id);
1823             return cstmt;
1824         }}, parameter);
1825 }
1826
1827 @Override
1828 public void update(int id, int age) {
1829     List<SqlParameter> parameters = Arrays.asList(
1830         new SqlParameter(Types.INTEGER), new SqlParameter(Types.INTEGER));
1831     this.jdbcTemplate.call(new CallableStatementCreator() {
1832         @Override
1833         public CallableStatement createCallableStatement(Connection conn) throws
1834             SQLException {
1835             CallableStatement cstmt = conn.prepareCall("{call sp_student_update(?, ?)}");
1836             cstmt.setInt(1, id);
1837             cstmt.setInt(2, age);
1838             return cstmt;
1839         }}, parameters);
1840 }
1841
1842 @Override
1843 public Student getStudent(int id) {
1844     List<SqlParameter> parameter = Arrays.asList(new SqlParameter(Types.INTEGER),
1845         new SqlOutParameter("v_name", Types.VARCHAR),
1846         new SqlOutParameter("v_age", Types.INTEGER));
1847     Map<String, Object> map = this.jdbcTemplate.call(new CallableStatementCreator() {
1848         @Override
1849         public CallableStatement createCallableStatement(Connection conn) throws
1850             SQLException {
1851             CallableStatement cstmt = conn.prepareCall("{call sp_student_select(?, ?, ?)}");
1852             cstmt.setInt(1, id);
1853             return cstmt;
1854         }}, parameter);
1855
1856     Student student = new Student();
1857     student.setId(id);
1858     student.setName((String)map.get("v_name"));
1859     student.setAge(Integer.parseInt(map.get("v_age").toString()));
1860     //Set<String> set = map.keySet();

```

```

1862         //System.out.println(set);    //[#update-count-1, v_name, v_age]
1863     return student;
1864 }
1865
1866 @Override
1867 public List<Student> listStudents() {
1868     List<SqlParameter> parameters = Arrays.asList(new SqlParameter(Types.NULL));
1869     Map<String, Object> map = this.jdbcTemplate.call(new CallableStatementCreator() {
1870         @Override
1871         public CallableStatement createCallableStatement(Connection conn) throws
1872             SQLException {
1873             CallableStatement cstmt = conn.prepareCall("{call sp_student_selectlist}");
1874             return cstmt;
1875         }}, parameters);
1876     //Set<String> set = map.keySet();
1877     //System.out.println(set);    //[#result-set-1, #update-count-1]
1878     return mapStudents(map, 1);
1879 }
1880
1881 private List<Student> mapStudents(Map<String, Object> out, int resultSetPosition){
1882     List<Student> students = new ArrayList<Student>();
1883     List<Map<String, Object>> results = (List<Map<String, Object>>)out.get("#result-set-1");
1884     results.forEach(s -> {
1885         Student student = new Student();
1886         student.setId((Integer)s.get("id"));
1887         student.setName((String)s.get("name"));
1888         student.setAge((Integer)s.get("age"));
1889         students.add(student);
1890     });
1891     return students;
1892 }
1893
1894

```

1895 15. beans.xml 생성

- 1896 1)resources > right-click > New > Other > Spring > Spring Bean Configuration File > Next
- 1897 2)File name : beans.xml
- 1898 3)Finish
- 1899 4)Namespaces tab > context check

```

1901 <?xml version="1.0" encoding="UTF-8"?>
1902 <beans xmlns="http://www.springframework.org/schema/beans"
1903     xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
1904     xmlns:context="http://www.springframework.org/schema/context"
1905     xsi:schemaLocation="http://www.springframework.org/schema/beans
1906         http://www.springframework.org/schema/beans/spring-beans.xsd">
1907
1908     <bean id="dataSource"
1909         class="org.springframework.jdbc.datasource.DriverManagerDataSource">
1910         <property name="driverClassName" value="${db.driverClass}" />
1911         <property name="url" value="${db.url}" />
1912         <property name="username" value="${db.username}" />
1913         <property name="password" value="${db.password}" />
1914     </bean>
1915
1916     <bean id="studentJdbcTemplate" class="com.example.StudentJDBCTemplate">
1917         <property name="dataSource" ref="dataSource" />
1918     </bean>
1919 </beans>

```

1920 16. MainClass Class 생성

- 1921 1)com.example > right-click > New > Class
- 1922 2)Name : MainClass
- 1923 3)Finish

```

1924
1925 package com.example;

```

```

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainClass {
    public static void main(String[] args) {
        ApplicationContext ctx = new ClassPathXmlApplicationContext("beans.xml");
        StudentJDBCTemplate temp =
            (StudentJDBCTemplate)ctx.getBean("studentJDBCTemplate");

        System.out.println("-----Records Creation-----");
        temp.create("Zara", 11);
        temp.create("Nuha", 2);
        temp.create("Ayan", 15);

        System.out.println("-----Listing Multiple Records-----");
        temp.listStudents().forEach(student -> {
            System.out.printf("ID : %d", student.getId());
            System.out.printf(", Name : %s", student.getName());
            System.out.printf(", Age : %d%n", student.getAge());
        });

        System.out.println("-----Search Record with ID = 1-----");
        Student student = temp.getStudent(1);
        System.out.printf("ID : %d", student.getId());
        System.out.printf(", Name : %s", student.getName());
        System.out.printf(", Age : %d%n", student.getAge());

        System.out.println("-----Delete Record with ID = 1-----");
        temp.delete(1);

        System.out.println("-----Updating Record with ID = 2-----");
        temp.update(2, 20);

        System.out.println("-----Listing Multiple Records-----");
        temp.listStudents().forEach(s -> {
            System.out.printf("ID : %d", s.getId());
            System.out.printf(", Name : %s", s.getName());
            System.out.printf(", Age : %d%n", s.getAge());
        });
    }
}

```