HOL : Using JdbcTemplate

Task1. CRUD of Spring JdbcTemplate

1. Create Table

```
    CREATE TABLE Student(
        id    INT   NOT NULL AUTO_INCREMENT,
        name  VARCHAR(20)  NOT NULL,
        age   INT NOT NULL,
        PRIMARY KEY(id)
    );
```

2. In Package Explorer > right-click > New > Java Project
    1)Project Name : JdbcTemplateDemo
    2)JRE
        -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
    3)Uncheck [Create module-info.java file]
    4)Next
    5)Finish

3. src > right-click > New > Package
    1)Name : com.example
    2)Finish

4. Java Project를 Spring Project로 변환
    1)JdbcTemplateDemo Project > right-click > Configure > Convert to Maven Project
        -Project : /JdbcTemplateDemo
        -Group Id : JdbcTemplateDemo
        -Artifact Id : JdbcTemplateDemo
        -version : 0.0.1-SNAPSHOT
        -Packaging : jar
        -Finish

    2)JdbcTemplateDemo Project > right-click > Spring > Add Spring Project Nature

    3)pom.xml file에 Spring Context Dependency 추가하기

```
        <version>0.0.1-SNAPSHOT</version>
        <dependencies>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>5.3.10</version>
            </dependency>
        </dependencies>
```

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

5. Lombok library 추가
    1)https://mvnrepository.com/에서 'lombok'으로 검색
    2)'Project Lombok' click
    3)1.18.20 click
    4)depency copy해서 pom.xml에 붙여넣기

```
        <dependencies>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>5.3.10</version>
            </dependency>
            <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
            <dependency>
                <groupId>org.projectlombok</groupId>
```

```xml
                <artifactId>lombok</artifactId>
                <version>1.18.20</version>
                <scope>provided</scope>
            </dependency>
        </dependencies>
```

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

6. pom.xml에 Jdbc Driver 설정하기
    1)Oracle 12C 이후 version일 경우, mvnrepository에서 oralce로 검색후, Ojdbc8 설치

```xml
        <dependency>
            <groupId>com.oracle.database.jdbc</groupId>
            <artifactId>ojdbc8</artifactId>
            <version>21.3.0.0</version>
        </dependency>
```

    2)Oracle 11g version일 경우
        -pom.xml에 붙여 넣고 Maven Install 하기

```xml
            <dependency>
                <groupId>com.oracle</groupId>
                <artifactId>ojdbc6</artifactId>
                <version>11.2</version>
            </dependency>
```

    3)MySQL의 경우, MySQL Connector/J로 들어가서

```xml
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.26</version>
        </dependency>
```

    4)MariaDB의 경우, MariaDB Java Client로 들어가서

```xml
        <dependency>
            <groupId>org.mariadb.jdbc</groupId>
            <artifactId>mariadb-java-client</artifactId>
            <version>2.7.4</version>
        </dependency>
```

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

7. Spring JDBC pom.xml에 추가하기
    1)pom.xml에 다음 코드 추가

```xml
        <dependency>
             <groupId>org.springframework</groupId>
             <artifactId>spring-jdbc</artifactId>
             <version>5.3.10</version>
        </dependency>
```

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

8. Student class 생성
    1)com.example > right-click > New > Class
    2)Name : Student
    3)Finish

```java
        package com.example;

        import lombok.AllArgsConstructor;
        import lombok.Getter;
        import lombok.NoArgsConstructor;
```

```
import lombok.Setter;

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
public class Student {
    private int id;
    private int age;
    private String name;
}
```

9. StudentDao interface 생성
    1)com.example > right-click > New > Interface
    2)Name : StudentDao
    3)Finish

```
package com.example;

import java.util.List;
import javax.sql.DataSource;

public class StudentDao {
    void setDataSource(DataSource ds);
    void create(String name, int age);
    void delete(int id);
    void updae(int id, int age);
    Student getStudent(int id);
    List<Student> listStudents();
}
```

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
db.url=jdbc:mysql://localhost:3306/world
db.username=root
db.password=javamysql
```

12. src/com.example.StudentMapper class 생성
    1)com.example > right-click > New > Class
    2)Name : StudentMapper
    3)Finish

```
package com.example;

import java.sql.ResultSet;
import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

public class StudentMapper implements RowMapper<Student>{
    public Student mapRow(ResultSet rs, int rowNum) throws SQLException{
```

```java
        Student student = new Student();
        student.setId(rs.getInt("id"));
        student.setName(rs.getString("name"));
        student.setAge(rs.getInt("age"));
        return student;
    }
}
```

13. StudentJDBCTemplate class 생성
   1)com.example > right-click > New > Class
   2)Name : StudentJDBCTemplate
   3)Finish

```java
package com.example;

import java.util.List;
import javax.sql.DataSource;

public class StudentJDBCTemplate implements StudentDao {
    private DataSource dataSource;
    private JdbcTemplate jdbcTemplate;

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

    @Override
    public void create(String name, int age){
        String sql = "INSERT INTO Student(name, age) VALUES (?, ?)";
        this.jdbcTemplate.update(sql, name, age);
        System.out.println("Insert Success");
    }

    @Override
    public Student getStudent(int id){
        String sql = "SELECT * FROM Student where id = ?";
        Student student = this.jdbcTemplate.queryForObject(sql, new StudentMapper(), id);
        return student;
    }

    @Override
    public List<Student> listStudents(){
        String sql = "SELECT * FROM Student";
        List<Student> students = this.jdbcTemplate.query(sql, new StudentMapper());
        return students;
    }

    @Override
    public void delete(int id){
        String sql = "DELETE FROM Student WHERE id = ?";
        this.jdbcTemplate.update(sql, id);
        System.out.println("Delete Success");
    }

    @Override
    public void updae(int id, int age){
        String sql = "UPDATE Student set age = ? WHERE id = ?";
        this.jdbcTemplate.update(sql, age, id);
        System.out.println("Update Success");
    }
}
```

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="usrname" 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. MainClasst 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.prinlnt("---------Listing Multple 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.prinlnt("---------Listing Multple 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());
            }
        }
    }
```

Task2. Example of Spring JdbcTemplate

1. Create Table

```
    CREATE TABLE Employee(
        id NUMBER(10),
        name VARCHAR2(100),
        salary NUMBER(10)
    );
```

2. In Package Explorer > right-click > New > Java Project
    1)Project Name : JdbcTemplateDemo1
    2)JRE
        -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
    3)Uncheck [Create module-info.java file]
    4)Next
    5)Finish

3. src > right-click > New > Package
    1)Name : com.example
    2)Finish

4. Java Project를 Spring Project로 변환
    1)JdbcTemplateDemo1 Project > right-click > Configure > Convert to Maven Project
        -Project : /JdbcTemplateDemo1
        -Group Id : JdbcTemplateDemo1
        -Artifact Id : JdbcTemplateDemo1
        -version : 0.0.1-SNAPSHOT
        -Packaging : jar
        -Finish

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

    3)pom.xml file에 Spring Context Dependency 추가하기

```
        <version>0.0.1-SNAPSHOT</version>
        <dependencies>
            <dependency>
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>5.3.10</version>
            </dependency>
        </dependencies>
```

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

5. Lombok library 추가
    1)https://mvnrepository.com/에서 'lombok'으로 검색
    2)'Project Lombok' click
    3)1.18.20 click
    4)depency copy해서 pom.xml에 붙여넣기

```
        <dependencies>
            <dependency>
```

```
                <groupId>org.springframework</groupId>
                <artifactId>spring-context</artifactId>
                <version>5.3.10</version>
            </dependency>
            <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
            <dependency>
                <groupId>org.projectlombok</groupId>
                <artifactId>lombok</artifactId>
                <version>1.18.20</version>
                <scope>provided</scope>
            </dependency>
        </dependencies>
```

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

6. pom.xml에 Jdbc Driver 설정하기

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

```
        <dependency>
            <groupId>com.oracle.database.jdbc</groupId>
            <artifactId>ojdbc8</artifactId>
            <version>21.3.0.0</version>
        </dependency>
```

    2)Oracle 11g version일 경우
        -pom.xml에 붙여 넣고 Maven Install 하기

```
            <dependency>
                <groupId>com.oracle</groupId>
                <artifactId>ojdbc6</artifactId>
                <version>11.2</version>
            </dependency>
```

    3)MySQL의 경우, MySQL Connector/J로 들어가서

```
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.26</version>
        </dependency>
```

    4)MariaDB의 경우, MariaDB Java Client로 들어가서

```
        <dependency>
            <groupId>org.mariadb.jdbc</groupId>
            <artifactId>mariadb-java-client</artifactId>
            <version>2.7.4</version>
        </dependency>
```

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

7. Spring JDBC pom.xml에 추가하기

    1)pom.xml에 다음 코드 추가

```
        <dependency>
             <groupId>org.springframework</groupId>
             <artifactId>spring-jdbc</artifactId>
             <version>5.3.10</version>
        </dependency>
```

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

8. Employee class 생성

    1)com.example > right-click > New > Class
    2)Name : Employee

3)Finish

```java
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

```java
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
```

```
    db.url=jdbc:mysql://localhost:3306/world
    db.username=root
    db.password=javamysql
```

12. Java Annotation 환경설정 파일 생성
  1)com.example > right-click > New > Class
  2)Name : ApplicationConfig
  3)Finish

```java
    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.support.PropertySourcesPlaceholderConfigurer;
    import org.springframework.core.io.ClassPathResource;
    import org.springframework.jdbc.core.JdbcTemplate;
    import org.springframework.jdbc.datasource.DriverManagerDataSource;

    @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 configurer = new
            PropertySourcesPlaceholderConfigurer();
            configurer.setLocation(new ClassPathResource("dbinfo.properties"));
            return configurer;
        }

        @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 JdbcTemplate jdbcTemplate() {
            JdbcTemplate template = new JdbcTemplate();
            template.setDataSource(this.dataSource());
            return template;
        }
    }
```

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

```java
    package com.example;
```

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

        public class MainClass {
            public static void main(String[] args) {
                ApplicationContext ctx=new AnnotationConfigApplicationContext(ApplicationConfig.class);

                EmployeeDao dao = (EmployeeDao)ctx.getBean("empDao");
                int status = dao.saveEmployee(new Employee(102,"Amit",35000));
                System.out.println("Insert Status = " + status);

                status = dao.updateEmployee(new Employee(102,"Sonoo",15000));
                System.out.println("Update Status = " + status);

                Employee e=new Employee();
                e.setId(102);
                status = dao.deleteEmployee(e);
                System.out.println("Delete Status = " + status);
            }
        }
```

    4)결과

```
        Insert Status = 1
        Update Status = 1
        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

```
    package com.example;

    import java.sql.PreparedStatement;
    import java.sql.SQLException;

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

    @Repository("empDao1")
    public class EmployeeDao1 {
        @Autowired
        private JdbcTemplate jdbcTemplate;

        public Boolean saveEmployeeByPreparedStatement(final Employee e){
            String query="INSERT INTO Employee VALUES(?,?,?)";
            return jdbcTemplate.execute(query,new PreparedStatementCallback<Boolean>(){
                @Override
                public Boolean doInPreparedStatement(PreparedStatement ps)
                        throws SQLException, DataAccessException {

                    ps.setInt(1,e.getId());
                    ps.setString(2,e.getName());
                    ps.setFloat(3,e.getSalary());

                    return ps.execute();

                }
            });
        }
    }
```

2. ApplicationConfig1 class 생성
    1)com.example.ApplicationConfig.java를 복사하여 붙여넣기
    2)이름변경 : ApplicationConfig1
    3)OK

```
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.JdbcTemplate;
import org.springframework.jdbc.datasource.DriverManagerDataSource;

@Configuration
@ComponentScan(basePackages = "com.example")
public class ApplicationConfig1 {
    @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 configurer = new
        PropertySourcesPlaceholderConfigurer();
        configurer.setLocation(new ClassPathResource("dbinfo.properties"));
        return configurer;
    }

    @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 JdbcTemplate jdbcTemplate() {
        JdbcTemplate template = new JdbcTemplate();
        template.setDataSource(this.dataSource());
        return template;
    }

    @Bean
    public EmployeeDao1 empDao1() {
        EmployeeDao1 empDao1 = new EmployeeDao1();
        return empDao1;
    }
}
```

    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" />
            <property name="password" value="tiger" />
        </bean>

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

        <bean id="empDao1" class="com.example.EmployeeDao1">
            <property name="jdbcTemplate" ref="jdbcTemplate" />
        </bean>
    </beans>
```

3. MainClass1 class 생성
    1)com.example.MainClass 복사하여 붙여넣기
    2)이름변경 : MainClass1
    3)OK

```
package com.example;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class Test {
    public static void main(String[] args) {
        ApplicationContext ctx=new AnnotationConfigApplicationContext(ApplicationConfig1.class);

        EmployeeDao1 dao1 = (EmployeeDao1)ctx.getBean("empDao1");
        dao1.saveEmployeeByPreparedStatement(new Employee(108,"Amit",35000));
    }
}
```

    4)applicationContext.xml을 사용할 경우 Test.java

```
    package com.example;

    import org.springframework.context.ApplicationContext;
    import org.springframework.context.support.ClassPathXmlApplicationContext;
    public class Test {

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

            EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");
            dao.saveEmployeeByPreparedStatement(new Employee(108,"Amit",35000));
        }
    }
```

Task4. ResultSetExtractor Example | Fetching Records by Spring JdbcTemplate
1. Employee.java 수정

```
package com.example;

import lombok.AllArgsConstructor;
import lombok.Getter;
```

```
    import lombok.NoArgsConstructor;
    import lombok.Setter;
    import lombok.ToString;

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

2. EmployeeDao2 class 생성
   1)com.example.EmployeeDao1를 복사하여 붙여넣기
   2)이름변경 : EmployeeDao2
   3)OK

```
        package com.example;

        import java.sql.ResultSet;
        import java.sql.SQLException;
        import java.util.ArrayList;
        import java.util.List;

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

        @Repository("empDao2")
        public class EmployeeDao2 {
            @Autowired
            private JdbcTemplate jdbcTemplate;

            public List<Employee> getAllEmployees(){
                 return jdbcTemplate.query("SELECT * FROM Employee", new
                 ResultSetExtractor<List<Employee>>(){
                    @Override
                     public List<Employee> extractData(ResultSet rs) throws SQLException,
                          DataAccessException {

                       List<Employee> list=new ArrayList<Employee>();
                       while(rs.next()){
                           Employee e=new Employee();
                           e.setId(rs.getInt(1));
                           e.setName(rs.getString(2));
                           e.setSalary(rs.getFloat(3));
                           list.add(e);
                       }
                       return list;
                    }
                    });
            }
        }
```

3. ApplicationConfig2 class 생성
   1)com.example.ApplicationConfig1.java를 복사하여 붙여넣기
   2)이름변경 : ApplicationConfig2
   3)OK

```
            ...
            @Bean
            public EmployeeDao2 empDao2() {
                EmployeeDao2 empDao2 = new EmployeeDao2();
                return empDao2;
            }
        }
```

    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" />
                <property name="password" value="tiger" />
            </bean>

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

            <bean id="empDao2" class="com.example.EmployeeDao2">
                <property name="jdbcTemplate" ref="jdbcTemplate" />
            </bean>

        </beans>
```

4. MainClass2 class 생성
    1)MainClass1.java copy하여 붙여기
    2)이름변경 : MainClass2
    3)OK

```
        package com.example;

        import org.springframework.context.ApplicationContext;
        import org.springframework.context.annotation.AnnotationConfigApplicationContext;

        public class MainClass2 {
            public static void main(String[] args) {
                ApplicationContext ctx=new AnnotationConfigApplicationContext(ApplicationConfig2.class);

                EmployeeDao2 dao2 = (EmployeeDao2)ctx.getBean("empDao2");
                dao2.getAllEmployees().forEach(emp -> System.out.println(emp));
            }
        }
```

    4)결과

```
        Employee(id=108, name=Amit, salary=35000.0)
```

    5)applicationContext.xml을 사용할 경우 Test.java

```
        package com.example;

        import java.util.List;

        import org.springframework.context.ApplicationContext;
        import org.springframework.context.support.ClassPathXmlApplicationContext;
        public class Test {

            public static void main(String[] args) {
                ApplicationContext ctx=new ClassPathXmlApplicationContext("applicationContext.xml");
            EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");
            List<Employee> list=dao.getAllEmployees();

            for(Employee e:list)
                System.out.println(e);

        }
    }
```

```
----------------------------------------------------
```
Task5. RowMapper Example | Fetching records by Spring JdbcTemplate
1. EmployeeDao3 class 생성
    1)com.example.EmployeeDao2를 복사하여 붙여넣기
    2)이름변경 : EmployeeDao3
    3)OK

```
    package com.example;

    import java.sql.ResultSet;
    import java.sql.SQLException;
    import java.util.List;

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

    @Repository("empDao3")
    public class EmployeeDao3 {
        @Autowired
        private JdbcTemplate jdbcTemplate;

        public List<Employee> getAllEmployeesRowMapper(){
            return jdbcTemplate.query("select * from employee",new RowMapper<Employee>(){
                @Override
                public Employee mapRow(ResultSet rs, int rownumber) throws SQLException {
                    Employee e=new Employee();
                    e.setId(rs.getInt(1));
                    e.setName(rs.getString(2));
                    e.setSalary(rs.getFloat(3));
                    return e;
                }
            });
        }
    }
```

2. ApplicationConfig3 class 생성
    1)com.example.ApplicationConfig2.java를 복사하여 붙여넣기
    2)이름변경 : ApplicationConfig3
    3)OK

```
    ...
    @Bean
    public EmployeeDao3 empDao3() {
        EmployeeDao3 empDao3 = new EmployeeDao3();
        return empDao3;
    }
```

    4)applicationContext.xml 를 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" />
            <property name="password" value="tiger" />
        </bean>

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

        <bean id="empDao3" class="com.example.EmployeeDao3">
            <property name="jdbcTemplate" ref="jdbcTemplate"></property>
        </bean>

    </beans>
```

3. MainClass3 class 생성
    1)MainClass2.java copy하여 붙여기
    2)이름변경 : MainClass3
    3)OK

```
package com.example;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainClass3 {
    public static void main(String[] args) {
        ApplicationContext ctx=new AnnotationConfigApplicationContext(ApplicationConfig3.class);

        EmployeeDao3 dao3 = (EmployeeDao3)ctx.getBean("empDao3");
        dao3.getAllEmployeesRowMapper().forEach(emp -> System.out.println(emp));
    }
}
```

    4)결과

```
Employee(id=108, name=Amit, salary=35000.0)
```

    5)applicationContext.xml을 사용할 경우 Test.java

```
package com.example;

import java.util.List;

import org.springframework.context.ApplicationContext;
import org.springframework.context.support.ClassPathXmlApplicationContext;
public class Test {

    public static void main(String[] args) {
        ApplicationContext ctx=new ClassPathXmlApplicationContext("applicationContext.xml");
        EmployeeDao dao=(EmployeeDao)ctx.getBean("edao");
        List<Employee> list=dao.getAllEmployeesRowMapper();

        for(Employee e:list)
            System.out.println(e);

    }
}
```

Task6. Spring NamedParameterJdbcTemplate Example

1. Create Table

```
CREATE TABLE Employee(
    id NUMBER(10),
    name VARCHAR2(100),
    salary NUMBER(10)
);
```

2. In Package Explorer > right-click > New > Java Project
    1)Project Name : NamedJdbcTemplateDemo
    2)JRE
        -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
    3)Uncheck [Create module-info.java file]
    4)Next
    5)Finish

3. src > right-click > New > Package
    1)Name : com.example
    2)Finish

4. Java Project를 Spring Project로 변환
    1)NamedJdbcTemplateDemo Project > right-click > Configure > Convert to Maven Project
        -Project : /NamedJdbcTemplateDemo
        -Group Id : NamedJdbcTemplateDemo
        -Artifact Id : NamedJdbcTemplateDemo
        -version : 0.0.1-SNAPSHOT
        -Packaging : jar
        -Finish

    2)NamedJdbcTemplateDemo Project > right-click > Spring > Add Spring Project Nature

    3)pom.xml file에 Spring Context Dependency 추가하기

```
<version>0.0.1-SNAPSHOT</version>
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.10</version>
    </dependency>
</dependencies>
```

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

5. Lombok library 추가
    1)https://mvnrepository.com/에서 'lombok'으로 검색
    2)'Project Lombok' click
    3)1.18.20 click
    4)depency copy해서 pom.xml에 붙여넣기

```
<dependency>
    <groupId>org.projectlombok</groupId>
    <artifactId>lombok</artifactId>
    <version>1.18.20</version>
    <scope>provided</scope>
</dependency>
```

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

6. pom.xml에 Jdbc Driver 설정하기
    1)Oracle 12C 이후 version일 경우, mvnrepository에서 oralce로 검색후, Ojdbc8 설치

```
        <dependency>
            <groupId>com.oracle.database.jdbc</groupId>
            <artifactId>ojdbc8</artifactId>
            <version>21.3.0.0</version>
        </dependency>
```

2)Oracle 11g version일 경우

-pom.xml에 붙여 넣고 Maven Install 하기

```
            <dependency>
                <groupId>com.oracle</groupId>
                <artifactId>ojdbc6</artifactId>
                <version>11.2</version>
            </dependency>
```

3)MySQL의 경우, MySQL Connector/J로 들어가서

```
        <dependency>
            <groupId>mysql</groupId>
            <artifactId>mysql-connector-java</artifactId>
            <version>8.0.26</version>
        </dependency>
```

4)MariaDB의 경우, MariaDB Java Client로 들어가서

```
        <dependency>
            <groupId>org.mariadb.jdbc</groupId>
            <artifactId>mariadb-java-client</artifactId>
            <version>2.7.4</version>
        </dependency>
```

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

[INFO] BUILD SUCCESS 확인

7. Spring JDBC pom.xml에 추가하기

1)pom.xml에 다음 코드 추가

```
        <dependency>
             <groupId>org.springframework</groupId>
             <artifactId>spring-jdbc</artifactId>
             <version>5.3.10</version>
        </dependency>
```

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

[INFO] BUILD SUCCESS 확인

8. Employee class 생성

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

```java
package com.example;

import lombok.AllArgsConstructor;
import lombok.Getter;

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

9. EmployeeDao class 생성

1)com.example > New > Class
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;
```

```java
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 configurer = new
        PropertySourcesPlaceholderConfigurer();
        configurer.setLocation(new ClassPathResource("dbinfo.properties"));
        return configurer;
    }

    @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
<?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" />
        <property name="password" value="tiger" />
    </bean>

    <bean id="jtemplate"
    class="org.springframework.jdbc.core.namedparam.NamedParameterJdbcTemplate">
        <constructor-arg ref="ds" />
    </bean>

    <bean id="empDao" class="com.example.EmployeeDao">
        <constructor-arg>
            <ref bean="jtemplate"/>
        </constructor-arg>
    </bean>

</beans>
```

13. MainClass class 생성
    1)com.example > New > Class
    2)Name : MainClass
    3)Finish

```
package com.example;

import org.springframework.context.ApplicationContext;
import org.springframework.context.annotation.AnnotationConfigApplicationContext;

public class MainClass {
    public static void main(String[] args) {
        ApplicationContext ctx = new AnnotationConfigApplicationContext(ApplicationConfig.class);

        EmployeeDao dao=(EmployeeDao)ctx.getBean("empDao");
        dao.save(new Employee(23,"sonoo",50000));

    }
}
```

    4)applicationContext.xml을 사용할 경우 Test.java

```
package com.example;

import org.springframework.beans.factory.BeanFactory;
import org.springframework.beans.factory.xml.XmlBeanFactory;
import org.springframework.core.io.ClassPathResource;
import org.springframework.core.io.Resource;

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

        Resource r=new ClassPathResource("applicationContext.xml");
        BeanFactory factory=new XmlBeanFactory(r);

        EmpDao dao=(EmpDao)factory.getBean("edao");
        dao.save(new Emp(23,"sonoo",50000));

    }
}
```

Task7. Calling Stored Procedure
1. Preparation Table & Stored Procedure

    1)Create Table

```
CREATE TABLE Student(
    id    INT   NOT NULL AUTO_INCREMENT,
    name  VARCHAR(20)  NOT NULL,
    age   INT NOT NULL,
    PRIMARY KEY(id)
);
```

2)Create Stored Procedure

```
    DROP PROCEDURE IF EXISTS sp_student_select;
    delimiter //
    CREATE PROCEDURE sp_student_select
    (
        IN      v_id   INT,
        OUT   v_name   VARCHAR(20),
        OUT   v_age  INT
    )
    BEGIN
        SELECT name, age  INTO v_name, v_age
        FROM Student
        WHERE id = v_id;
    END; //
    delimiter ;

    DROP PROCEDURE IF EXISTS sp_student_selectlist;
    delimiter //
    CREATE PROCEDURE sp_student_selectlist()
    BEGIN
        SELECT *
        FROM Student;
    END; //
    delimiter ;

    DROP PROCEDURE IF EXISTS sp_student_insert;
    delimiter //
    CREATE PROCEDURE sp_student_insert
    (
        IN      v_name    VARCHAR(20),
        IN      v_age      INT
    )
    BEGIN
        INSERT INTO Student(name, age)
        VALUES(v_name, v_age);
        COMMIT;
    END; //
    delimiter ;

    DROP PROCEDURE IF EXISTS sp_student_update;
    delimiter //
    CREATE PROCEDURE sp_student_update
    (
        IN      v_id   INT,
        IN      v_age  INT
    )
    BEGIN
        UPDATE Student SET age = v_age
        WHERE id = v_id;
        COMMIT;
    END; //
    delimiter ;

    DROP PROCEDURE IF EXISTS sp_student_delete;
    delimiter //
    CREATE PROCEDURE sp_student_delete
    (
        IN      v_id   INT
    )
    BEGIN
        DELETE FROM Student
        WHERE id = v_id;
        COMMIT;
    END; //
    delimiter ;
```

2. In Package Explorer > right-click > New > Java Project
   1)Project Name : JdbcTemplateDemo2
   2)JRE
      -Select [Use default JRE 'jdk-11.0.12' and workspace compiler preferences]
   3)Uncheck [Create module-info.java file]
   4)Next
   5)Finish

3. src > right-click > New > Package
   1)Name : com.example
   2)Finish

4. Java Project를 Spring Project로 변환
   1)JdbcTemplateDemo2 Project > right-click > Configure > Convert to Maven Project
      -Project : /JdbcTemplateDemo2
      -Group Id : JdbcTemplateDemo2
      -Artifact Id : JdbcTemplateDemo2
      -version : 0.0.1-SNAPSHOT
      -Packaging : jar
      -Finish

   2)JdbcTemplateDemo2 Project > right-click > Spring > Add Spring Project Nature

   3)pom.xml file에 Spring Context Dependency 추가하기

```
<version>0.0.1-SNAPSHOT</version>
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.10</version>
    </dependency>
</dependencies>
```

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

5. Lombok library 추가
   1)https://mvnrepository.com/에서 'lombok'으로 검색
   2)'Project Lombok' click
   3)1.18.20 click
   4)depency copy해서 pom.xml에 붙여넣기

```
<dependencies>
    <dependency>
        <groupId>org.springframework</groupId>
        <artifactId>spring-context</artifactId>
        <version>5.3.10</version>
    </dependency>
    <!-- https://mvnrepository.com/artifact/org.projectlombok/lombok -->
    <dependency>
        <groupId>org.projectlombok</groupId>
        <artifactId>lombok</artifactId>
        <version>1.18.20</version>
        <scope>provided</scope>
    </dependency>
</dependencies>
```

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

6. pom.xml에 Jdbc Driver 설정하기
1)Oracle 12C 이후 version일 경우, mvnrepository에서 oralce로 검색후, Ojdbc8 설치

```
<dependency>
    <groupId>com.oracle.database.jdbc</groupId>
    <artifactId>ojdbc8</artifactId>
    <version>21.3.0.0</version>
</dependency>
```

2)Oracle 11g version일 경우
-pom.xml에 붙여 넣고 Maven Install 하기

```
<dependency>
    <groupId>com.oracle</groupId>
    <artifactId>ojdbc6</artifactId>
    <version>11.2</version>
</dependency>
```

3)MySQL의 경우, MySQL Connector/J로 들어가서

```
<dependency>
    <groupId>mysql</groupId>
    <artifactId>mysql-connector-java</artifactId>
    <version>8.0.26</version>
</dependency>
```

4)MariaDB의 경우, MariaDB Java Client로 들어가서

```
<dependency>
    <groupId>org.mariadb.jdbc</groupId>
    <artifactId>mariadb-java-client</artifactId>
    <version>2.7.4</version>
</dependency>
```

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

7. Spring JDBC pom.xml에 추가하기
1)pom.xml에 다음 코드 추가

```
<dependency>
     <groupId>org.springframework</groupId>
     <artifactId>spring-jdbc</artifactId>
     <version>5.3.10</version>
</dependency>
```

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

8. Student class 생성
1)com.example > right-click > New > Class
2)Name : Student
3)Finish

```
package com.example;

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

@Getter
@Setter
@AllArgsConstructor
@NoArgsConstructor
public class Student {
    private int id;
    private String name;
```

```
        private int age;
    }
```

9. StudentDao interface 생성
    1)com.example > right-click > New > Interface
    2)Name : StudentDao
    3)Finish

```
    package com.example;

    import java.util.List;
    import javax.sql.DataSource;

    public class StudentDao {
        void setDataSource(DataSource ds);
        void create(String name, int age);
        void delete(int id);
        void updae(int id, int age);
        Student getStudent(int id);
        List<Student> listStudents();
    }
```

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
        db.url=jdbc:mysql://localhost:3306/world
        db.username=root
        db.password=javamysql
```

12. src/com.example.StudentMapper class 생성
    1)com.example > right-click > New > Class
    2)Name : StudentMapper
    3)Finish

```
    package com.example;

    import java.sql.ResultSet;
    import java.sql.SQLException;

    import org.springframework.jdbc.core.RowMapper;

    public class StudentMapper implements RowMapper<Student>{
        public Student mapRow(ResultSet rs, int rowNum) throws SQLException{
            Student student = new Student();
            student.setId(rs.getInt("id"));
            student.setName(rs.getString("name"));
            student.setAge(rs.getInt("age"));
            return student;
        }
    }
```

13. StudentJDBCTemplate class 생성 ==> SimpleJdbcCall 사용하기
  1)com.example > right-click > New > Class
  2)Name : StudentJDBCTemplate
  3)Finish

```java
    package com.example;

    import java.sql.Types;
    import java.util.ArrayList;
    import java.util.List;
    import java.util.Map;
    import java.util.Set;

    import javax.sql.DataSource;

    import org.springframework.jdbc.core.JdbcTemplate;
    import org.springframework.jdbc.core.SqlOutParameter;
    import org.springframework.jdbc.core.namedparam.MapSqlParameterSource;
    import org.springframework.jdbc.core.namedparam.SqlParameterSource;
    import org.springframework.jdbc.core.simple.SimpleJdbcCall;

    public class StudentJDBCTemplate implements StudentDao {
        private DataSource dataSource;
        private JdbcTemplate jdbcTemplate;

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

        @Override
        public void create(String name, int age) {
            SimpleJdbcCall jdbcCall =
                    new SimpleJdbcCall(this.dataSource).withProcedureName("sp_student_insert");
            SqlParameterSource in = new MapSqlParameterSource().addValue("v_name", name)
                    .addValue("v_age", age);
            jdbcCall.execute(in);
        }

        @Override
        public void delete(int id) {
            SimpleJdbcCall jdbcCall =
                    new SimpleJdbcCall(this.dataSource).withProcedureName("sp_student_delete");
            SqlParameterSource in = new MapSqlParameterSource().addValue("v_id", id);
            jdbcCall.execute(in);
        }

        @Override
        public void update(int id, int age) {
            SimpleJdbcCall jdbcCall =
                    new SimpleJdbcCall(this.dataSource).withProcedureName("sp_student_update");
            SqlParameterSource in = new MapSqlParameterSource().addValue("v_id", id)
                    .addValue("v_age", age);
            jdbcCall.execute(in);
        }

        @Override
        public Student getStudent(int id) {
            SimpleJdbcCall jdbcCall =
                    new SimpleJdbcCall(this.dataSource).withProcedureName("sp_student_select")
                    .declareParameters(new SqlOutParameter("v_name", Types.VARCHAR),
                                       new SqlOutParameter("v_age", Types.INTEGER));
            SqlParameterSource in = new MapSqlParameterSource().addValue("v_id", id);
            Map<String, Object> map = jdbcCall.execute(in);

            Student student = new Student();
            student.setId(id);
            student.setName((String)map.get("v_name"));
            student.setAge((Integer)map.get("v_age"));
            //student.setAge(Integer.parseInt(map.get("v_age").toString()));
            //Set<String> set = map.keySet();
            //System.out.println(set);    //[#update-count-1, v_name, v_age]
            return student;
        }

        @Override
        public List<Student> listStudents() {
            SimpleJdbcCall jdbcCall =
                    new SimpleJdbcCall(this.dataSource).withProcedureName("sp_student_selectlist");
            SqlParameterSource in = new MapSqlParameterSource();
            Map<String, Object> map = jdbcCall.execute(in);
            return mapStudents(map, 1);
        }

        private List<Student> mapStudents(Map<String, Object> out, int resultSetPosition){
            List<Student> students = new ArrayList<Student>();
            List<Map<String, Object>> results = (List<Map<String, Object>>)out.get("#result-set-1");
            results.forEach(s -> {
                Student student = new Student();
                student.setId((Integer)s.get("id"));
                student.setName((String)s.get("name"));
                student.setAge((Integer)s.get("age"));
                students.add(student);
            });
            return students;
        }

    }
```

14. StudentJDBCTemplate class 생성 ==> CallableStatementCreator 사용하기
    1)com.example > right-click > New > Class
    2)Name : StudentJDBCTemplate1
    3)Finish

```
package com.example;

import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.SQLException;
import java.sql.Types;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.List;
import java.util.Map;

import javax.sql.DataSource;

import org.springframework.jdbc.core.CallableStatementCreator;
import org.springframework.jdbc.core.JdbcTemplate;
import org.springframework.jdbc.core.SqlOutParameter;
import org.springframework.jdbc.core.SqlParameter;

public class StudentJDBCTemplate implements StudentDao {
    private DataSource dataSource;
    private JdbcTemplate jdbcTemplate;

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

```

```java
        @Override
        public void create(String name, int age) {
            List<SqlParameter> parameters = Arrays.asList(
                    new SqlParameter(Types.VARCHAR), new SqlParameter(Types.INTEGER));
            this.jdbcTemplate.call(new CallableStatementCreator() {
                @Override
                public CallableStatement createCallableStatement(Connection conn) throws
                SQLException {
                    CallableStatement cstmt = conn.prepareCall("{call sp_student_insert(?, ?)}");
                    cstmt.setString(1, name);
                    cstmt.setInt(2, age);
                    return cstmt;
                }}, parameters);
        }

        @Override
        public void delete(int id) {
            List<SqlParameter> parameter = Arrays.asList(new SqlParameter(Types.INTEGER));
            this.jdbcTemplate.call(new CallableStatementCreator() {
                @Override
                public CallableStatement createCallableStatement(Connection con) throws
                SQLException {
                    CallableStatement cstmt = con.prepareCall("{call sp_student_delete(?)}");
                    cstmt.setInt(1, id);
                    return cstmt;
                }
            }, parameter);
        }

        @Override
        public void update(int id, int age) {
            List<SqlParameter> parameters = Arrays.asList(
                    new SqlParameter(Types.INTEGER), new SqlParameter(Types.INTEGER));
            this.jdbcTemplate.call(new CallableStatementCreator() {
                @Override
                public CallableStatement createCallableStatement(Connection conn) throws
                SQLException {
                    CallableStatement cstmt = conn.prepareCall("{call sp_student_update(?, ?)}");
                    cstmt.setInt(1, id);
                    cstmt.setInt(2, age);
                    return cstmt;
                }
            }, parameters);
        }

        @Override
        public Student getStudent(int id) {
            List<SqlParameter> parameter = Arrays.asList(new SqlParameter(Types.INTEGER),
                    new SqlOutParameter("v_name", Types.VARCHAR),
                    new SqlOutParameter("v_age", Types.INTEGER));
            Map<String, Object> map = this.jdbcTemplate.call(new CallableStatementCreator() {

                @Override
                public CallableStatement createCallableStatement(Connection conn) throws
                SQLException {
                    CallableStatement cstmt = conn.prepareCall("{call sp_student_select(?, ?, ?)}");
                    cstmt.setInt(1, id);
                    return cstmt;
                }

            }, parameter);

            Student student = new Student();
            student.setId(id);
            student.setName((String)map.get("v_name"));
            student.setAge(Integer.parseInt(map.get("v_age").toString()));
            //Set<String> set = map.keySet();
            //System.out.println(set);    //[#update-count-1, v_name, v_age]
            return student;
        }

        @Override
        public List<Student> listStudents() {
            List<SqlParameter> parameters = Arrays.asList(new SqlParameter(Types.NULL));
            Map<String, Object> map = this.jdbcTemplate.call(new CallableStatementCreator() {
                @Override
                public CallableStatement createCallableStatement(Connection conn) throws
                SQLException {
                    CallableStatement cstmt = conn.prepareCall("{call sp_student_selectlist}");
                    return cstmt;
                }}, parameters);
            //Set<String> set = map.keySet();
            //System.out.println(set);    //[#result-set-1, #update-count-1]
            return mapStudents(map, 1);
        }

        private List<Student> mapStudents(Map<String, Object> out, int resultSetPosition){
            List<Student> students = new ArrayList<Student>();
            List<Map<String, Object>> results = (List<Map<String, Object>>)out.get("#result-set-1");
            results.forEach(s -> {
                Student student = new Student();
                student.setId((Integer)s.get("id"));
                student.setName((String)s.get("name"));
                student.setAge((Integer)s.get("age"));
                students.add(student);
            });
            return students;
        }
    }
```

15. 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="usrname" value="${db.username}" />
            <property name="password" value="${db.password}" />
        </bean>

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

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

```
    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());
        });
    }
}
```