Spring JDBC:

---------

Spring JDBC is designed on the top of JDBC API.

Which is used to implify the operations of JDBC.

Spring JDBC Advantages:

1) Connection Management taken care by Spring container.

Not required to create connection object and close connection object by programmer.

2) template based program reduces the coding lines.

3) Supports passing parameters as var-args method.

Insted of passing as an object[], we can pass individual values for operation method.

POJI-POJO Desing Pattern:

-------------------------

Spring supports POJI-POJO Programming design pattern for layer implementation code.

Here,

POJI : indicates Plane Old Java Interface. Which provides the details of operations to be performed on model object for business class.

POJO: It is an implemented class for POJI. It contains implementation logic related to business operation. A POJI can contain more then one POJO also.

But at a time It referes only one POJO.

Model Bean/Model Class/Model:

-------------------------------

A class Which represents a database table or UI Form to store data or to transfer data at application level.

It is also know as DTO(Data Transfer Object).

JdbcTemplate: This is a class given from Spring API.

Which holds an reference of DataSource(I).

DataSource: It represent a connection object details, it is an interfece given from javax.sql package.

DriverManagerDataSource: It is an implimentation class of DataSource(I). Which provides a connection object to perform the JDBC operations.

--------------------------------------

Steps To Write Spring JDBC Program:

1)Java Code

2) XML Code

1) Java Code:

i)Design A model Bean (Employee.java), which represents

database table.

ii) Design an operational Interface to perform operations on model bean(IEmployeeDao.java).

iii) Provide an implementation(EmployeeJdbcDaoImpl.java) to Design Interface (IEmployee.java).

iV) Add a dependency of JdbcTemplate To Implemented class.

And provide mutators.

----------------------------------------------------

ex:

Logic to store a record using Spring JDBC:

public void addEmployeeDetails(Employee emp) {

String sql="insert into employee values(?,?,?)";

int count=template.update(sql, emp.getEmpId(), emp.getEmpName(),emp.getEmpSal());

}

--------------------------------------------

2) Xml Code:

---------------

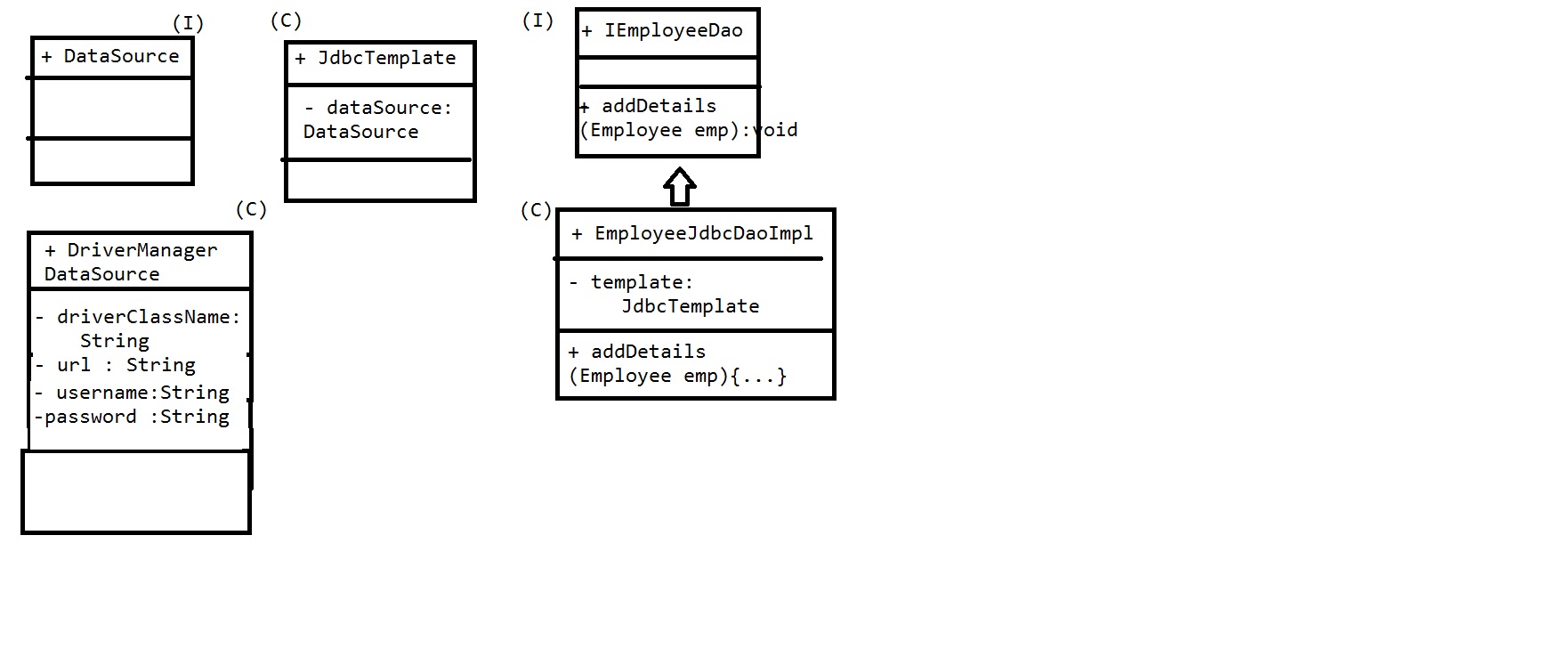
Here We have to configure 3 java classes in below order in spring configuration file.

i) DriverManagerDataSource (uname,password,url,driver-class)

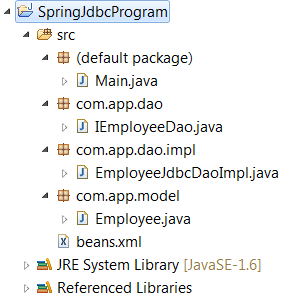
ii) JdbcTemplate (datasource obj)

iii) EmployeeJdbcDaoImpl (templateObj).

Dependency Structure is:



Programs:



Employee.java

**package** com.app.model;

**public** **class** Employee {

**private** **int** empId;

**private** String empName;

**private** **double** empSal;

**public** Employee() {

**super**();

}

**public** **int** getEmpId() {

**return** empId;

}

**public** **void** setEmpId(**int** empId) {

**this**.empId = empId;

}

**public** String getEmpName() {

**return** empName;

}

**public** **void** setEmpName(String empName) {

**this**.empName = empName;

}

**public** **double** getEmpSal() {

**return** empSal;

}

**public** **void** setEmpSal(**double** empSal) {

**this**.empSal = empSal;

}

@Override

**public** String toString() {

**return** "Employee [empId=" + empId + ", empName=" + empName

+ ", empSal=" + empSal + "]";

}

}

IEmployeeDao.java

**package** com.app.dao;

**import** com.app.model.Employee;

**public** **interface** IEmployeeDao {

**public** **void** addEmployeeDetails(Employee emp);

}

EmployeeJdbcDaoImpl.java

**package** com.app.dao.impl;

**import** org.springframework.jdbc.core.JdbcTemplate;

**import** com.app.dao.IEmployeeDao;

**import** com.app.model.Employee;

**public** **class** EmployeeJdbcDaoImpl **implements** IEmployeeDao {

**private** JdbcTemplate template;

**public** **void** setTemplate(JdbcTemplate template) {

**this**.template = template;

}

@Override

**public** **void** addEmployeeDetails(Employee emp) {

String sql="insert into employee values(?,?,?)";

**int** count=template.update(sql, emp.getEmpId(),emp.getEmpName(),emp.getEmpSal());

System.*out*.println("Record(s) Inserted:"+count);

}

}

Beans.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:context=*"http://www.springframework.org/schema/context"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans*

*http://www.springframework.org/schema/beans/spring-beans.xsd*

*http://www.springframework.org/schema/context*

*http://www.springframework.org/schema/context/spring-context.xsd*

*"*>

<bean class=*"org.springframework.jdbc.datasource.DriverManagerDataSource"* name=*"dataSourceObj"*>

<property name=*"driverClassName"* value=*"com.mysql.jdbc.Driver"*/>

<property name=*"url"* value=*"jdbc:mysql://localhost:3306/test"*/>

<property name=*"username"* value=*"root"*/>

<property name=*"password"* value=*"root"*/>

</bean>

<bean class=*"org.springframework.jdbc.core.JdbcTemplate"* name=*"templateObj"*>

<property name=*"dataSource"*>

<ref bean=*"dataSourceObj"*/>

</property>

</bean>

<bean class=*"com.app.dao.impl.EmployeeJdbcDaoImpl"* name=*"empDaoObj"*>

<property name=*"template"*>

<ref bean=*"templateObj"*/>

</property>

</bean>

</beans>

Main.java

**import** org.springframework.context.ApplicationContext;

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

**import** com.app.dao.IEmployeeDao;

**import** com.app.dao.impl.EmployeeJdbcDaoImpl;

**import** com.app.model.Employee;

**public** **class** Main {

**public** **static** **void** main(String[] args) {

ApplicationContext context=**new** ClassPathXmlApplicationContext("beans.xml");

IEmployeeDao dao=context.getBean("empDaoObj", EmployeeJdbcDaoImpl.**class**);

Employee emp=**new** Employee();

emp.setEmpId(101);

emp.setEmpName("AJ");

emp.setEmpSal(200.36);

dao.addEmployeeDetails(emp);

System.*out*.println("Done");

}

}

Note:

Add database related Jar file also. Ex: my-sql-connector.jar

