Spring :

Employee Class:

**package** com.deere.spring;

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

**private** **int** id;

**private** String fname;

**private** String lname;

**private** Address addr;

**public** Employee() {

System.***out***.println("inside constructor");

}

**public** Employee(**int** id, String fname, String lname, Address addr) {

**super**();

**this**.id = id;

**this**.fname = fname;

**this**.lname = lname;

**this**.addr = addr;

}

**public** **int** getId() {

**return** id;

}

**public** **void** setId(**int** id) {

**this**.id = id;

System.***out***.println("inside set id");

}

**public** String getFname() {

**return** fname;

}

**public** **void** setFname(String fname) {

**this**.fname = fname;

System.***out***.println("inside set first name");

}

**public** String getLname() {

**return** lname;

}

**public** **void** setLname(String lname) {

**this**.lname = lname;

System.***out***.println("inside set last name");

}

**public** Address getAddr() {

**return** addr;

}

**public** **void** setAddr(Address addr) {

**this**.addr = addr;

System.***out***.println("inside set addr");

}

@Override

**public** String toString() {

**return** "Employee [id=" + id + ", fname=" + fname + ", lname=" + lname + ", addr=" + addr.toString() + "]";

}

}

Address Class:

**package** com.deere.spring;

**public** **class** Address {

**private** String city;

**private** **int** zip;

**public** Address() {

**super**();

System.***out***.println("inside address constructor");

}

**public** Address(String city, **int** zip) {

**super**();

**this**.city = city;

**this**.zip = zip;

System.***out***.println("Inside parameterized constructor of address");

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

System.***out***.println("inside city");

}

**public** **int** getZip() {

**return** zip;

}

**public** **void** setZip(**int** zip) {

**this**.zip = zip;

System.***out***.println("inside zip");

}

@Override

**public** String toString() {

**return** "Address [city=" + city + ", zip=" + zip + "]";

}

}

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"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<bean id = *"e1"* class=*"com.deere.spring.Employee"* scope=*"prototype"*>

<property name=*"id"* value=*"101"*></property>

<property name=*"fname"* value=*"Dipti"*></property>

<property name=*"lname"* value=*"Jain"*></property>

<property name=*"addr"* ref=*"a1"*></property>

</bean>

<bean id = *"a1"* class=*"com.deere.spring.Address"*>

<property name=*"city"* value=*"Pune"*></property>

<property name=*"zip"* value=*"411013"*></property>

</bean>

</beans>

Main class :

**package** com.deere.spring;

**import** org.omg.CORBA.portable.ApplicationException;

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

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

**public** **class** MainClass {

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

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

Employee emp = context.getBean("e1", Employee.**class**);

System.***out***.println(emp);

Employee emp2 = context.getBean("e1", Employee.**class**);

System.***out***.println(emp2);

}

}

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<bean id = *"e1"* class=*"com.deere.spring.Employee"*>

<constructor-arg index=*"0"* value=*"102"*></constructor-arg>

<constructor-arg index=*"1"* value=*"DJ"*></constructor-arg>

<constructor-arg index=*"2"* value=*"Jain2"*></constructor-arg>

<constructor-arg index=*"3"* ref=*"a1"*></constructor-arg>

<!-- <property name="id" value="101"></property>

<property name="fname" value="Dipti"></property>

<property name="lname" value="Jain"></property>

<property name="addr" ref="a1"></property> -->

</bean>

<bean id = *"a1"* class=*"com.deere.spring.Address"*>

<constructor-arg index=*"0"* value=*"Pune"*></constructor-arg>

<constructor-arg index=*"1"* value=*"411012"*></constructor-arg>

<property name=*"city"* value=*"Mumbai"*></property>

<property name=*"zip"* value=*"411013"*></property>

</bean>

</beans>

Spring Lifecycle :

Employee Class :

package com.deere.spring;

import java.util.Properties;

import org.springframework.beans.BeansException;

import org.springframework.beans.PropertyValue;

import org.springframework.beans.PropertyValues;

import org.springframework.beans.factory.BeanNameAware;

import org.springframework.beans.factory.InitializingBean;

import org.springframework.context.ApplicationContext;

import org.springframework.context.ApplicationContextAware;

public class Employee implements BeanNameAware, ApplicationContextAware, InitializingBean{

private int id;

private String fname;

private String lname;

private Address addr;

private ApplicationContext ctx;

public Employee() {

System.out.println("inside constructor");

}

public Employee(int id, String fname, String lname, Address addr) {

super();

this.id = id;

this.fname = fname;

this.lname = lname;

this.addr = addr;

System.out.println("Inside Employee parameterized constructor");

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

System.out.println("inside set id");

}

public String getFname() {

return fname;

}

public void setFname(String fname) {

this.fname = fname;

System.out.println("inside set first name");

}

public String getLname() {

return lname;

}

public void setLname(String lname) {

this.lname = lname;

System.out.println("inside set last name");

}

public Address getAddr() {

return addr;

}

public void setAddr(Address addr) {

this.addr = addr;

System.out.println("inside set addr");

}

@Override

public void setBeanName(String arg0) {

System.out.println("Inside setBean Name - " +arg0);

}

@Override

public void setApplicationContext(ApplicationContext ctx) throws BeansException {

System.out.println("Inside aet application context - " +ctx);

//this.addr = ctx.getBean("a2", Address.class);

this.ctx = ctx;

}

@Override

public String toString() {

return "Employee [id=" + id + ", fname=" + fname + ", lname=" + lname + ", addr=" + addr.toString() + "]";

}

@Override

public void afterPropertiesSet() throws Exception {

System.out.println("Inside After property set");

this.fname = "Ashish";

this.lname = "Tiwari";

this.addr = (Address) this.ctx.getBean("a2");

}

public void doinit() {

System.out.println("Inside do init method");

this.fname = "Ashish2";

this.lname = "Tiwari2";

this.addr = (Address) this.ctx.getBean("a2");

}

public void destroy() {

System.out.println("Inside destroy method.");

}

}

Address Class :

**package** com.deere.spring;

**public** **class** Address {

**private** String city;

**private** **int** zip;

**public** Address() {

**super**();

System.***out***.println("inside address constructor");

}

**public** Address(String city, **int** zip) {

**super**();

**this**.city = city;

**this**.zip = zip;

System.***out***.println("Inside parameterized constructor of address");

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

System.***out***.println("inside city");

}

**public** **int** getZip() {

**return** zip;

}

**public** **void** setZip(**int** zip) {

**this**.zip = zip;

System.***out***.println("inside zip");

}

@Override

**public** String toString() {

**return** "Address [city=" + city + ", zip=" + zip + "]";

}

}

Main class :

package com.deere.spring;

import org.omg.CORBA.portable.ApplicationException;

import org.springframework.context.ApplicationContext;

import org.springframework.context.ConfigurableApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainClass {

public static void main(String[] args) {

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

Employee emp = context.getBean("e1", Employee.class);

System.out.println(emp);

Employee emp2 = context.getBean("e1", Employee.class);

System.out.println(emp2);

ConfigurableApplicationContext conctx = (ConfigurableApplicationContext) context;

conctx.close();

}

}

Xml file :

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<bean id = *"e1"* class=*"com.deere.spring.Employee"* init-method=*"doinit"* destroy-method=*"destroy"*>

<constructor-arg index=*"0"* value=*"102"*></constructor-arg>

<constructor-arg index=*"1"* value=*"DJ"*></constructor-arg>

<constructor-arg index=*"2"* value=*"Jain2"*></constructor-arg>

<constructor-arg index=*"3"* ref=*"a1"*></constructor-arg>

<property name=*"id"* value=*"101"*></property>

<property name=*"fname"* value=*"Dipti"*></property>

<property name=*"lname"* value=*"Jain"*></property>

<property name=*"addr"* ref=*"a1"*></property>

</bean>

<bean id = *"a1"* class=*"com.deere.spring.Address"*>

<constructor-arg index=*"0"* value=*"Pune"*></constructor-arg>

<constructor-arg index=*"1"* value=*"411012"*></constructor-arg>

<property name=*"city"* value=*"Mumbai"*></property>

<property name=*"zip"* value=*"411013"*></property>

</bean>

<bean id = *"a2"* class=*"com.deere.spring.Address"*>

<!-- <constructor-arg index="0" value="Pune"></constructor-arg>

<constructor-arg index="1" value="411012"></constructor-arg> -->

<property name=*"city"* value=*"ujjain"*></property>

<property name=*"zip"* value=*"411010"*></property>

</bean>

</beans>

Event Handling :

HRDept :

package com.deere;

import org.springframework.beans.BeansException;

import org.springframework.context.ApplicationContext;

import org.springframework.context.ApplicationContextAware;

public class HRDepart implements ApplicationContextAware {

private ApplicationContext ctx;

public void recruitEmployee() {

System.out.println("Inside RECRUIT employee.");

ctx.publishEvent(new EmployeeRecruitedEvent(this, "Dipti"));

}

@Override

public void setApplicationContext(ApplicationContext context) throws BeansException {

System.out.println("inside APPLICATIONCONTEXT");

this.ctx = context;

}

}

Finance Dept. :

**package** com.deere;

**import** org.springframework.context.ApplicationListener;

**public** **class** FinanceDept **implements** ApplicationListener<EmployeeRecruitedEvent>{

@Override

**public** **void** onApplicationEvent(EmployeeRecruitedEvent event) {

String name = event.getName();

System.***out***.println("Finance Dept started processing of the employee: " +name);

}

}

Tech Dept :

**package** com.deere;

**import** org.springframework.context.ApplicationListener;

**public** **class** TechDept **implements** ApplicationListener<EmployeeRecruitedEvent>{

@Override

**public** **void** onApplicationEvent(EmployeeRecruitedEvent event) {

String name = event.getName();

System.***out***.println("Tech Dept started processing of the employee: " +name);

}

}

EmployeeRecruitedEvent:

**package** com.deere;

**import** org.springframework.context.ApplicationEvent;

**public** **class** EmployeeRecruitedEvent **extends** ApplicationEvent {

**private** String name;

**public** EmployeeRecruitedEvent(Object source, String name) {

**super**(source);

**this**.name = name;

}

**public** String getName() {

**return** name;

}

}

Events.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"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<bean id=*"hr"* class=*"com.deere.HRDepart"*>

</bean>

<bean id=*"tech"* class=*"com.deere.TechDept"*>

</bean>

<bean id=*"fin"* class=*"com.deere.FinanceDept"*>

</bean>

</beans>

ReportGeneration :

ReportService class:

package com.deere;

import java.io.FileNotFoundException;

import java.io.PrintWriter;

public class ReportService {

private String filepath;

private PrintWriter writer;

public ReportService() {

System.out.println("Inside default constructor of ReportService.");

}

public String getFilepath() {

return filepath;

}

public void setFilepath(String filepath) {

this.filepath = filepath;

System.out.println("Inside setFilepath");

}

public void startup() {

System.out.println("Inside startup");

try {

writer = new PrintWriter(filepath);

} catch (FileNotFoundException e) {

e.printStackTrace();

}

}

public void generateReport() {

System.out.println("Inside generate report");

writer.print("The app is generating report into file");

}

public void cleanup() {

System.out.println("clean up");

writer.close();

}

}

Main Class :

package com.deere;

import org.springframework.context.ApplicationContext;

import org.springframework.context.ConfigurableApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainClass {

public static void main(String[] args) {

ApplicationContext context = new ClassPathXmlApplicationContext("reports.xml");

ReportService rs = context.getBean("rs", ReportService.class);

rs.generateReport();

ConfigurableApplicationContext ctxc = (ConfigurableApplicationContext) context;

ctxc.close();

}

}

Reports.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:util=*"http://www.springframework.org/schema/util"*

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

*http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.3.xsd"*>

<bean id = *"rs"* class=*"com.deere.ReportService"* init-method=*"startup"* destroy-method=*"cleanup"*>

<property name=*"filepath"* value=*"C:\Users\adm\Documents\DJ57741\dj.txt"*></property>

</bean>

<util:properties id = *""*></util:properties>

</beans>

Above code with filename passed in config file :

<?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:util=*"http://www.springframework.org/schema/util"*

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

*http://www.springframework.org/schema/util http://www.springframework.org/schema/util/spring-util-4.3.xsd"*>

<bean id = *"rs"* class=*"com.deere.ReportService"* init-method=*"startup"* destroy-method=*"cleanup"*>

<!-- <property name="filepath" value="C:\Users\adm\Documents\DJ57741\dj.txt"></property> -->

<property name=*"filepath"* value=*"#{prop['filename']}"*></property>

</bean>

<util:properties id = *"prop"* location=*"classpath:config.properties"*></util:properties>

</beans>

Cofig file :

filename=C:\\Users\\adm\\Documents\\DJ57741\\abc.txt

Autowire byName and byType : Above classes are same.

byType :

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<bean id = *"e1"* class=*"com.deere.spring.Employee"* init-method=*"doinit"* destroy-method=*"destroy"* autowire=*"byType"*>

<!-- <constructor-arg index="0" value="102"></constructor-arg>

<constructor-arg index="1" value="DJ"></constructor-arg>

<constructor-arg index="2" value="Jain2"></constructor-arg>

<constructor-arg index="3" ref="a1"></constructor-arg>

-->

<property name=*"id"* value=*"101"*></property>

<property name=*"fname"* value=*"Dipti"*></property>

<property name=*"lname"* value=*"Jain"*></property>

<!-- <property name="addr" ref="a1"></property> -->

</bean>

<bean id = *"Address"* class=*"com.deere.spring.Address"*>

<constructor-arg index=*"0"* value=*"Pune"*></constructor-arg>

<constructor-arg index=*"1"* value=*"411012"*></constructor-arg>

<property name=*"city"* value=*"Mumbai"*></property>

<property name=*"zip"* value=*"411013"*></property>

</bean>

<!-- <bean id = "a2" class="com.deere.spring.Address">

<constructor-arg index="0" value="Pune"></constructor-arg>

<constructor-arg index="1" value="411012"></constructor-arg>

<property name="city" value="ujjain"></property>

<property name="zip" value="411010"></property>

</bean>

-->

</beans>

<?xml version=*"1.0"* encoding=*"UTF-8"*?>

<beans xmlns=*"http://www.springframework.org/schema/beans"*

xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://www.springframework.org/schema/beans http://www.springframework.org/schema/beans/spring-beans.xsd"*>

<bean id = *"e1"* class=*"com.deere.spring.Employee"* init-method=*"doinit"* destroy-method=*"destroy"* autowire=*"byType"*>

<!-- <constructor-arg index="0" value="102"></constructor-arg>

<constructor-arg index="1" value="DJ"></constructor-arg>

<constructor-arg index="2" value="Jain2"></constructor-arg>

<constructor-arg index="3" ref="a1"></constructor-arg>

-->

<property name=*"id"* value=*"101"*></property>

<property name=*"fname"* value=*"Dipti"*></property>

<property name=*"lname"* value=*"Jain"*></property>

<!-- <property name="addr" ref="a1"></property> -->

</bean>

<bean id = *"Address"* class=*"com.deere.spring.Address"*>

<constructor-arg index=*"0"* value=*"Pune"*></constructor-arg>

<constructor-arg index=*"1"* value=*"411012"*></constructor-arg>

<property name=*"city"* value=*"Mumbai"*></property>

<property name=*"zip"* value=*"411013"*></property>

</bean>

<!-- <bean id = "a2" class="com.deere.spring.Address">

<constructor-arg index="0" value="Pune"></constructor-arg>

<constructor-arg index="1" value="411012"></constructor-arg>

<property name="city" value="ujjain"></property>

<property name="zip" value="411010"></property>

</bean>

-->

</beans>

Autowired with annotation :

package com.deere.spring;

import java.util.Properties;

import org.springframework.beans.BeansException;

import org.springframework.beans.PropertyValue;

import org.springframework.beans.PropertyValues;

import org.springframework.beans.factory.BeanNameAware;

import org.springframework.beans.factory.InitializingBean;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.context.ApplicationContext;

import org.springframework.context.ApplicationContextAware;

public class Employee implements BeanNameAware, ApplicationContextAware, InitializingBean{

private int id;

private String fname;

private String lname;

@Autowired

private Address addr;

private ApplicationContext ctx;

public Employee() {

System.out.println("inside constructor");

}

public Employee(int id, String fname, String lname, Address addr) {

super();

this.id = id;

this.fname = fname;

this.lname = lname;

this.addr = addr;

System.out.println("Inside Employee parameterized constructor");

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

System.out.println("inside set id");

}

public String getFname() {

return fname;

}

public void setFname(String fname) {

this.fname = fname;

System.out.println("inside set first name");

}

public String getLname() {

return lname;

}

public void setLname(String lname) {

this.lname = lname;

System.out.println("inside set last name");

}

public Address getAddr() {

return addr;

}

public void setAddr(Address addr) {

this.addr = addr;

System.out.println("inside set addr");

}

@Override

public void setBeanName(String arg0) {

System.out.println("Inside setBean Name - " +arg0);

}

@Override

public void setApplicationContext(ApplicationContext ctx) throws BeansException {

System.out.println("Inside aet application context - " +ctx);

//this.addr = ctx.getBean("a2", Address.class);

this.ctx = ctx;

}

@Override

public String toString() {

return "Employee [id=" + id + ", fname=" + fname + ", lname=" + lname + ", addr=" + addr.toString() + "]";

}

@Override

public void afterPropertiesSet() throws Exception {

System.out.println("Inside After property set");

this.fname = "Ashish";

this.lname = "Tiwari";

//this.addr = (Address) this.ctx.getBean("a2");

}

public void doinit() {

System.out.println("Inside do init method");

this.fname = "Ashish2";

this.lname = "Tiwari2";

//this.addr = (Address) this.ctx.getBean("a2");

}

public void destroy() {

System.out.println("Inside destroy method.");

}

}

<?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-4.3.xsd"*>

<bean id = *"e1"* class=*"com.deere.spring.Employee"* init-method=*"doinit"* destroy-method=*"destroy"* >

<!-- <constructor-arg index="0" value="102"></constructor-arg>

<constructor-arg index="1" value="DJ"></constructor-arg>

<constructor-arg index="2" value="Jain2"></constructor-arg>

<constructor-arg index="3" ref="a1"></constructor-arg>

-->

<property name=*"id"* value=*"101"*></property>

<property name=*"fname"* value=*"Dipti"*></property>

<property name=*"lname"* value=*"Jain"*></property>

<!-- <property name="addr" ref="a1"></property> -->

</bean>

<bean id = *"Address"* class=*"com.deere.spring.Address"*>

<constructor-arg index=*"0"* value=*"Pune"*></constructor-arg>

<constructor-arg index=*"1"* value=*"411012"*></constructor-arg>

<property name=*"city"* value=*"Mumbai"*></property>

<property name=*"zip"* value=*"411013"*></property>

</bean>

<!-- <bean id = "a2" class="com.deere.spring.Address">

<constructor-arg index="0" value="Pune"></constructor-arg>

<constructor-arg index="1" value="411012"></constructor-arg>

<property name="city" value="ujjain"></property>

<property name="zip" value="411010"></property>

</bean>

-->

<context:annotation-config></context:annotation-config>

<context:component-scan base-package="com.deere"></context:component-scan> // This is for base class.

</beans>

With only annotation :

package com.deere.spring;

import org.omg.CORBA.portable.ApplicationException;

import org.springframework.context.ApplicationContext;

import org.springframework.context.ConfigurableApplicationContext;

import org.springframework.context.annotation.AnnotationConfigApplicationContext;

import org.springframework.context.support.ClassPathXmlApplicationContext;

public class MainClass {

public static void main(String[] args) {

ApplicationContext context = new AnnotationConfigApplicationContext(MyConfig.class);

// new ClassPathXmlApplicationContext("beans.xml");

Employee emp = context.getBean("e1", Employee.class);

System.out.println(emp);

Employee emp2 = context.getBean("e1", Employee.class);

System.out.println(emp2);

ConfigurableApplicationContext conctx = (ConfigurableApplicationContext) context;

conctx.close();

}

}

Employee :

package com.deere.spring;

import java.util.Properties;

import org.springframework.beans.BeansException;

import org.springframework.beans.PropertyValue;

import org.springframework.beans.PropertyValues;

import org.springframework.beans.factory.BeanNameAware;

import org.springframework.beans.factory.InitializingBean;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.beans.factory.annotation.Value;

import org.springframework.context.ApplicationContext;

import org.springframework.context.ApplicationContextAware;

import org.springframework.stereotype.Component;

@Component(value = "e1")

public class Employee implements BeanNameAware, ApplicationContextAware{

@Value("111")

private int id;

@Value("Ayansh")

private String fname;

@Value("Tiwari")

private String lname;

@Autowired

private Address addr;

private ApplicationContext ctx;

public Employee() {

System.out.println("inside constructor");

}

public Employee(int id, String fname, String lname, Address addr) {

super();

this.id = id;

this.fname = fname;

this.lname = lname;

this.addr = addr;

System.out.println("Inside Employee parameterized constructor");

}

public int getId() {

return id;

}

public void setId(int id) {

this.id = id;

System.out.println("inside set id");

}

public String getFname() {

return fname;

}

public void setFname(String fname) {

this.fname = fname;

System.out.println("inside set first name");

}

public String getLname() {

return lname;

}

public void setLname(String lname) {

this.lname = lname;

System.out.println("inside set last name");

}

public Address getAddr() {

return addr;

}

public void setAddr(Address addr) {

this.addr = addr;

System.out.println("inside set addr");

}

@Override

public void setBeanName(String arg0) {

System.out.println("Inside setBean Name - " +arg0);

}

@Override

public void setApplicationContext(ApplicationContext ctx) throws BeansException {

System.out.println("Inside aet application context - " +ctx);

//this.addr = ctx.getBean("a2", Address.class);

this.ctx = ctx;

}

@Override

public String toString() {

return "Employee [id=" + id + ", fname=" + fname + ", lname=" + lname + ", addr=" + addr.toString() + "]";

}

/\* @Override

public void afterPropertiesSet() throws Exception {

System.out.println("Inside After property set");

this.fname = "Ashish";

this.lname = "Tiwari";

//this.addr = (Address) this.ctx.getBean("a2");

}

\*/

public void doinit() {

System.out.println("Inside do init method");

this.fname = "Ashish2";

this.lname = "Tiwari2";

//this.addr = (Address) this.ctx.getBean("a2");

}

public void destroy() {

System.out.println("Inside destroy method.");

}

}

Address :

**package** com.deere.spring;

**import** org.springframework.beans.factory.annotation.Value;

**import** org.springframework.stereotype.Component;

@Component(value = "a1")

**public** **class** Address {

@Value("Indore")

**private** String city;

@Value("456010")

**private** **int** zip;

**public** Address() {

**super**();

System.***out***.println("inside address constructor");

}

**public** Address(String city, **int** zip) {

**super**();

**this**.city = city;

**this**.zip = zip;

System.***out***.println("Inside parameterized constructor of address");

}

**public** String getCity() {

**return** city;

}

**public** **void** setCity(String city) {

**this**.city = city;

System.***out***.println("inside city");

}

**public** **int** getZip() {

**return** zip;

}

**public** **void** setZip(**int** zip) {

**this**.zip = zip;

System.***out***.println("inside zip");

}

@Override

**public** String toString() {

**return** "Address [city=" + city + ", zip=" + zip + "]";

}

}

**package** com.deere.spring;

**import** org.springframework.context.annotation.ComponentScan;

**import** org.springframework.context.annotation.Configuration;

@Configuration

@ComponentScan(basePackages="com.deere")

**public** **class** MyConfig {

}

Aspect Oriented Programming :

**package** com.jd.service;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Component;

**import** com.jd.data.Idao;

**import** com.jd.model.Employee;

@Component("service")

**public** **class** EmployeeService **implements** Iservice {

@Autowired

**private** Idao dao;

@Override

**public** **void** saveEmployees(List<Employee> list) {

//System.out.println("Inside saveEmployee method in service layer");

**for**(Employee emp : list) {

dao.saveEmployee(emp);

}

}

@Override

**public** **void** modifyEmployees(List<Employee> list) {

}

}

**package** com.jd.service;

**import** java.util.List;

**import** com.jd.model.Employee;

**public** **interface** Iservice {

**void** saveEmployees(List<Employee> list);

**void** modifyEmployees(List<Employee> list);

}

**package** com.jd.data;

**import** com.jd.model.Employee;

**public** **interface** Idao {

**void** saveEmployee(Employee e);

**void** modifyEmployee(Employee e);

}

**package** com.jd.data;

**import** org.springframework.stereotype.Component;

**import** com.jd.model.Employee;

@Component("dao")

**public** **class** EmployeeDAO **implements** Idao {

@Override

**public** **void** saveEmployee(Employee e) {

//System.out.println("Inside saveEmployee method of data layer.");

}

@Override

**public** **void** modifyEmployee(Employee e) {

//System.out.println("Inside modifyEmployee method of data layer.");

}

}

<?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"*

xmlns:aop=*"http://www.springframework.org/schema/aop"*

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

*http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context-4.3.xsd*

*http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.3.xsd"*>

<context:annotation-config></context:annotation-config>

<context:component-scan base-package=*"com.jd.\*"*></context:component-scan>

<aop:aspectj-autoproxy></aop:aspectj-autoproxy>

</beans>

**import** java.util.ArrayList;

**import** java.util.List;

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

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

**import** com.jd.model.Employee;

**import** com.jd.service.EmployeeService;

**import** com.jd.service.Iservice;

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

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

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

//EmployeeService serv = context.getBean("service", EmployeeService.class);

Iservice serv = context.getBean("service", Iservice.**class**);

List<Employee> list = **new** ArrayList<>();

list.add(**new** Employee(10, "Dipti", "Jain"));

list.add(**new** Employee(7, "Ayansh", "Tiwari"));

serv.saveEmployees(list);

}

}

**package** com.jd.aop;

**import** org.aspectj.lang.JoinPoint;

**import** org.aspectj.lang.ProceedingJoinPoint;

**import** org.aspectj.lang.annotation.AfterReturning;

**import** org.aspectj.lang.annotation.AfterThrowing;

**import** org.aspectj.lang.annotation.Around;

**import** org.aspectj.lang.annotation.Aspect;

**import** org.aspectj.lang.annotation.Before;

**import** org.aspectj.lang.annotation.Pointcut;

**import** org.springframework.stereotype.Component;

@Aspect

@Component

**public** **class** loggingAspect {

@Pointcut(value = "execution(\* \*.save\*(..))")

**public** **void** dummy() {

}

@Before(value = "dummy()") // instead of repeating expression we are calling method.

**public** **void** logBefore(JoinPoint jp) {

System.***out***.println("Inside method:"+ jp.getSignature().getName());

}

@AfterReturning(value = "dummy()", returning="r") // return only when business component executed successfully.

**public** **void** logAfter(JoinPoint jp, Object r) {

System.***out***.println("Exiting from the method."+ jp.getSignature().getName());

}

@AfterThrowing(value = "dummy()", throwing="e")

**public** **void** logError(Exception e) {

System.***out***.println("Inside exception method.");

}

@Around(value = "dummy()")

**public** Object logAround(ProceedingJoinPoint pjp) {

System.***out***.println("Inside ProceedingJoinPoint method");

Object returnval = **null**;

**try** {

System.***out***.println("Inside ProceedingJoinPoint method : Before of Around");

returnval = pjp.proceed();

System.***out***.println("Inside ProceedingJoinPoint method : After of Around");

} **catch** (Throwable e) {

System.***out***.println("Inside ProceedingJoinPoint method : Afterthrowing of Around");

e.printStackTrace();

}

**return** returnval;

}

}

SpringAOP with Database :

**Main class :**

**import** java.util.ArrayList;

**import** java.util.List;

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

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

**import** com.jd.data.EmployeeDAO;

**import** com.jd.data.Idao;

**import** com.jd.model.Employee;

**import** com.jd.service.EmployeeService;

**import** com.jd.service.Iservice;

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

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

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

//EmployeeService serv = context.getBean("service", EmployeeService.class);

Iservice serv = context.getBean("service", Iservice.**class**);

List<Employee> list = **new** ArrayList<>();

list.add(**new** Employee(10, "DiptiJ", "Jain"));

list.add(**new** Employee(7, "Ayanshh", "Tiwari"));

list.add(**new** Employee(5, "Ayuuuu", "Tiwari"));

list.add(**new** Employee(1, "Ashishh", "Tiwari"));

//serv.saveEmployees(list);

//serv.modifyEmployees(list);

Idao edao = context.getBean("dao", Idao.**class**);

List<Employee> emplist = edao.getAllEmployee();

**for** (Employee e : emplist) {

System.***out***.println(e.toString());

}

System.***out***.println(edao.getEmployeebyID(10));

System.***out***.println(edao.getEmployeeCount());

}

}

Logging Aspect :

**package** com.jd.aop;

**import** org.aspectj.lang.JoinPoint;

**import** org.aspectj.lang.ProceedingJoinPoint;

**import** org.aspectj.lang.annotation.AfterReturning;

**import** org.aspectj.lang.annotation.AfterThrowing;

**import** org.aspectj.lang.annotation.Around;

**import** org.aspectj.lang.annotation.Aspect;

**import** org.aspectj.lang.annotation.Before;

**import** org.aspectj.lang.annotation.Pointcut;

**import** org.springframework.stereotype.Component;

@Aspect

@Component

**public** **class** loggingAspect {

@Pointcut(value = "execution(\* \*.save\*(..))")

**public** **void** dummy() {

}

@Before(value = "dummy()") // instead of repeating expression we are calling method.

**public** **void** logBefore(JoinPoint jp) {

System.***out***.println("Inside method:"+ jp.getSignature().getName());

}

@AfterReturning(value = "dummy()", returning="r") // return only when business component executed successfully.

**public** **void** logAfter(JoinPoint jp, Object r) {

System.***out***.println("Exiting from the method."+ jp.getSignature().getName());

}

@AfterThrowing(value = "dummy()", throwing="e")

**public** **void** logError(Exception e) {

System.***out***.println("Inside exception method.");

}

@Around(value = "dummy()")

**public** Object logAround(ProceedingJoinPoint pjp) {

System.***out***.println("Inside ProceedingJoinPoint method");

Object returnval = **null**;

**try** {

System.***out***.println("Inside ProceedingJoinPoint method : Before of Around");

returnval = pjp.proceed();

System.***out***.println("Inside ProceedingJoinPoint method : After of Around");

} **catch** (Throwable e) {

System.***out***.println("Inside ProceedingJoinPoint method : Afterthrowing of Around");

e.printStackTrace();

}

**return** returnval;

}

}

Employee Class :

**package** com.jd.model;

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

**private** **int** eid;

**private** String fname;

**private** String lname;

**public** Employee() {

**super**();

// **TODO** Auto-generated constructor stub

}

**public** Employee(**int** eid, String fname, String lname) {

**super**();

**this**.eid = eid;

**this**.fname = fname;

**this**.lname = lname;

}

**public** **int** getEid() {

**return** eid;

}

**public** **void** setEid(**int** eid) {

**this**.eid = eid;

}

/\*public int getId() {

return eid;

}

public void setId(int id) {

this.eid = id;

}\*/

**public** String getFname() {

**return** fname;

}

**public** **void** setFname(String fname) {

**this**.fname = fname;

}

**public** String getLname() {

**return** lname;

}

**public** **void** setLname(String lname) {

**this**.lname = lname;

}

@Override

**public** String toString() {

**return** "Employee [eid=" + eid + ", fname=" + fname + ", lname=" + lname + "]";

}

}

Employee Service :

**package** com.jd.service;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

**import** org.springframework.stereotype.Component;

**import** com.jd.data.Idao;

**import** com.jd.model.Employee;

@Component("service")

**public** **class** EmployeeService **implements** Iservice {

@Autowired

**private** Idao dao;

@Override

**public** **void** saveEmployees(List<Employee> list) {

//System.out.println("Inside saveEmployee method in service layer");

**for**(Employee emp : list) {

dao.saveEmployee(emp);

}

}

@Override

**public** **void** modifyEmployees(List<Employee> list) {

}

}

Iservice :

**package** com.jd.service;

**import** java.util.List;

**import** com.jd.model.Employee;

**public** **interface** Iservice {

**void** saveEmployees(List<Employee> list);

**void** modifyEmployees(List<Employee> list);

}

Employee Dao :

**package** com.jd.data;

**import** java.util.List;

**import** org.springframework.beans.factory.annotation.Autowired;

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

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

**import** org.springframework.stereotype.Component;

**import** com.jd.model.Employee;

@Component("dao")

**public** **class** EmployeeDAO **implements** Idao {

@Autowired

**private** JdbcTemplate template;

@Override

**public** **void** saveEmployee(Employee e) {

//System.out.println("Inside saveEmployee method of data layer.");

String sql = "insert into Employee value(?,?,?)";

template.update(sql,e.getEid(),e.getFname(),e.getLname());

}

@Override

**public** **void** modifyEmployee(Employee e) {

//System.out.println("Inside modifyEmployee method of data layer.");

String sql = "update Employee set fname=?, lname=? where Id=?";

template.update(sql,e.getFname(),e.getLname(), e.getEid());

}

@Override

**public** List<Employee> getAllEmployee() {

// **TODO** Auto-generated method stub

String sql = "select \* from Employee";

//return template.query(sql, new BeanPropertyRowMapper<Employee>(Employee.class));

// below row we have writtern if I have mismatching variable name and DB column name. And that is why we create new class "EmployeeRowMapper" to map.

**return** template.query(sql, **new** EmployeeRowMapper());

}

@Override

**public** Employee getEmployeebyID(**int** empid) {

String sql = "select \* from Employee where Id=?";

**return** template.queryForObject(sql, **new** Object[] {empid}, **new** EmployeeRowMapper());

}

@Override

**public** String getEmployeeNamebyID(**int** empid) {

String sql = "select fname from Employee where Id=?";

**return** template.queryForObject(sql, String.**class**, empid);

}

@Override

**public** **int** getEmployeeCount() {

String sql = "select count(\*) from Employee";

**return** template.queryForObject(sql, Integer.**class**);

}

}

Idao :

**package** com.jd.data;

**import** java.util.List;

**import** com.jd.model.Employee;

**public** **interface** Idao {

**void** saveEmployee(Employee e);

**void** modifyEmployee(Employee e);

List<Employee> getAllEmployee();

Employee getEmployeebyID(**int** empid);

String getEmployeeNamebyID(**int** empid);

**int** getEmployeeCount();

}

Row mapper customized class :

package com.jd.data;

import java.sql.ResultSet;

import java.sql.SQLException;

import org.springframework.jdbc.core.RowMapper;

import com.jd.model.Employee;

// this class is for custom row mapper.

public class EmployeeRowMapper implements RowMapper<Employee> {

@Override

public Employee mapRow(ResultSet rs, int rownumber) throws SQLException {

System.out.println("Inside mapRow");

int id = rs.getInt("id");

String fn = rs.getString("fname");

String ln = rs.getString("lname");

Employee e = new Employee(id, fn, ln);

return e;

}

}

Aop 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"*

xmlns:aop=*"http://www.springframework.org/schema/aop"*

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

*http://www.springframework.org/schema/context https://www.springframework.org/schema/context/spring-context-4.3.xsd*

*http://www.springframework.org/schema/aop http://www.springframework.org/schema/aop/spring-aop-4.3.xsd"*>

<context:annotation-config></context:annotation-config>

<context:component-scan base-package=*"com.jd.\*"*></context:component-scan>

<aop:aspectj-autoproxy></aop:aspectj-autoproxy>

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

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

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

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

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

</bean>

<bean id=*"template"* class=*"org.springframework.jdbc.core.JdbcTemplate"*>

<property name=*"dataSource"* ref=*"ds"*></property>

</bean>

</beans>

Above employee program with Transactions, propogation : only changes is in Employee DAO and EmployeeService

package com.jd.data;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.jdbc.core.BeanPropertyRowMapper;

import org.springframework.jdbc.core.JdbcTemplate;

import org.springframework.stereotype.Component;

import org.springframework.transaction.PlatformTransactionManager;

import org.springframework.transaction.annotation.Propagation;

import org.springframework.transaction.annotation.Transactional;

import com.jd.model.Employee;

@Component("dao")

public class EmployeeDAO implements Idao {

@Autowired

private JdbcTemplate template;

private PlatformTransactionManager transactionManager;

@Override

//@Transactional(propagation = Propagation.SUPPORTS)

@Transactional(propagation = Propagation.REQUIRES\_NEW)

public void saveEmployee(Employee e) {

//System.out.println("Inside saveEmployee method of data layer.");

String sql = "insert into Employee value(?,?,?)";

template.update(sql,e.getEid(),e.getFname(),e.getLname());

}

@Override

public void modifyEmployee(Employee e) {

//System.out.println("Inside modifyEmployee method of data layer.");

String sql = "update Employee set fname=?, lname=? where Id=?";

template.update(sql,e.getFname(),e.getLname(), e.getEid());

}

@Override

public List<Employee> getAllEmployee() {

// TODO Auto-generated method stub

String sql = "select \* from Employee";

//return template.query(sql, new BeanPropertyRowMapper<Employee>(Employee.class));

// below row we have writtern if I have mismatching variable name and DB column name. And that is why we create new class "EmployeeRowMapper" to map.

return template.query(sql, new EmployeeRowMapper());

}

@Override

public Employee getEmployeebyID(int empid) {

String sql = "select \* from Employee where Id=?";

return template.queryForObject(sql, new Object[] {empid}, new EmployeeRowMapper());

}

@Override

public String getEmployeeNamebyID(int empid) {

String sql = "select fname from Employee where Id=?";

return template.queryForObject(sql, String.class, empid);

}

@Override

public int getEmployeeCount() {

String sql = "select count(\*) from Employee";

return template.queryForObject(sql, Integer.class);

}

}

package com.jd.service;

import java.util.List;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.stereotype.Component;

import org.springframework.transaction.PlatformTransactionManager;

import org.springframework.transaction.annotation.Isolation;

import org.springframework.transaction.annotation.Propagation;

import org.springframework.transaction.annotation.Transactional;

import com.jd.data.Idao;

import com.jd.model.Employee;

@Component("service")

public class EmployeeService implements Iservice {

@Autowired

private Idao dao;

@Autowired

private PlatformTransactionManager transactionManager;

@Override

@Transactional(propagation = Propagation.REQUIRED,rollbackFor = Exception.class, isolation=Isolation.DEFAULT)

public void saveEmployees(List<Employee> list) {

//System.out.println("Inside saveEmployee method in service layer");

for(Employee emp : list) {

dao.saveEmployee(emp);

}

}

@Override

public void modifyEmployees(List<Employee> list) {

}

}