**Transaction manager**

Create a maven project

Pom.xml add the dependencies.

Copy the hexa.properties in resources folder

Ceate packeage hexa.client and add the config file

Create package com.hexa.entity and create class Account

Create package com.hexa.dao and create an interface AccountDao

Create an implementation class AccImplDao

Create com.hexa.service and have an interface and implementation in it

<project xmlns=*"http://maven.apache.org/POM/4.0.0"* xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"*

xsi:schemaLocation=*"http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd"*>

<modelVersion>4.0.0</modelVersion>

<groupId>com.hexa</groupId>

<artifactId>SpringHibernateTx</artifactId>

<version>0.0.1-SNAPSHOT</version>

<packaging>jar</packaging>

<name>SpringHibernateTx</name>

<url>http://maven.apache.org</url>

<properties>

<project.build.sourceEncoding>UTF-8</project.build.sourceEncoding>

<maven.compiler.source>1.8</maven.compiler.source>

<maven.compiler.target>1.8</maven.compiler.target>

</properties>

<dependencies>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-context</artifactId>

<version>5.0.1.RELEASE</version>

</dependency>

<dependency>

<groupId>org.springframework</groupId>

<artifactId>spring-orm</artifactId>

<version>5.0.1.RELEASE</version>

</dependency>

<dependency>

<groupId>org.hibernate</groupId>

<artifactId>hibernate-core</artifactId>

<version>5.1.0.Final</version>

</dependency>

<dependency>

<groupId>mysql</groupId>

<artifactId>mysql-connector-java</artifactId>

<version>5.1.6</version>

</dependency>

</dependencies>

</project>

driver=com.mysql.jdbc.Driver

url=jdbc:mysql://localhost:3306/test

uname=root

pwd=root

showsql=true

formatsql=true

dialect=org.hibernate.dialect.MySQLDialect

hbm2ddl=auto

package com.hexa.entity;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.Id;

import javax.persistence.Table;

@Entity

@Table(name="account")

public class Account {

@Id

@Column(name="aid")

private int accId;

@Column(name="cname")

private String custName;

@Column(name="bal")

private double bal;

**public** Account() {

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

}

**public** Account(**int** accId, String custName, **double** bal) {

**super**();

**this**.accId = accId;

**this**.custName = custName;

**this**.bal = bal;

}

Client Code :

**package** com.hexa.client;

**import** java.util.ArrayList;

**import** java.util.List;

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

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

**import** com.hexa.entity.Account;

**import** com.hexa.service.ISer;

**public** **class** TestAddAccount {

**private** **static** ApplicationContext *ctx*;

**static** {

*ctx* = **new** AnnotationConfigApplicationContext(HibernateConfig.**class**);

}

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

ISer ser = *ctx*.getBean("myser", ISer.**class**);

Account acc1 = **new** Account(1005,"Rita",25000);

Account acc2 = **new** Account(1004,"Peter",15000);

List<Account> accLst = **new** ArrayList<>();

accLst.add(acc1);

accLst.add(acc2);

**int** res = ser.addAccounts(accLst);

**if**(res > 0 ) {

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

}

}

}

public int getAccId() {

return accId;

}

public void setAccId(int accId) {

this.accId = accId;

}

public String getCustName() {

return custName;

}

public void setCustName(String custName) {

this.custName = custName;

}

public double getBal() {

return bal;

}

public void setBal(double bal) {

this.bal = bal;

}

@Override

public String toString() {

return accId + " " + custName + " " + bal;

}

}

***HibernateConfig.class file***

package com.hexa.client;

import java.util.Properties;

import javax.sql.DataSource;

import org.hibernate.SessionFactory;

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

import org.springframework.context.annotation.Bean;

import org.springframework.context.annotation.ComponentScan;

import org.springframework.context.annotation.Configuration;

import org.springframework.context.annotation.PropertySource;

import org.springframework.core.env.Environment;

import org.springframework.jdbc.datasource.DriverManagerDataSource;

import org.springframework.orm.hibernate5.HibernateTransactionManager;

import org.springframework.orm.hibernate5.LocalSessionFactoryBean;

import org.springframework.transaction.annotation.EnableTransactionManagement;

@Configuration

@EnableTransactionManagement

@PropertySource(value="classpath:hexa.properties")

@ComponentScan("com.hexa")

public class HibernateConfig {

@Autowired

private Environment env;

@Bean

public DriverManagerDataSource getdataSource(){

DriverManagerDataSource ds = new DriverManagerDataSource();

ds.setDriverClassName(env.getProperty("driver"));

ds.setUrl(env.getProperty("url"));

ds.setUsername(env.getProperty("uname"));

ds.setPassword(env.getProperty("pwd"));

return ds;

}

@Bean

@Autowired

public LocalSessionFactoryBean getSessionFactory(DriverManagerDataSource ds){

System.out.println(ds.getUsername());

LocalSessionFactoryBean fac = new LocalSessionFactoryBean();

fac.setDataSource(ds);

fac.setHibernateProperties(getProperties());

fac.setPackagesToScan(new String[]{"com.hexa.entity"});

return fac;

}

public Properties getProperties(){

Properties props = new Properties();

props.setProperty("hibernate.dialect", env.getProperty("dialect"));

props.setProperty("hibernate.show\_sql", env.getProperty("showsql"));

props.setProperty("hibernate.format\_sql", env.getProperty("formatsql"));

return props;

}

@Bean

@Autowired

public HibernateTransactionManager getTxmanger(SessionFactory sfac){

HibernateTransactionManager tx = new HibernateTransactionManager();

tx.setSessionFactory(sfac);

return tx;

}

}

**package** com.hexa.dao;

**import** org.hibernate.Session;

**import** org.hibernate.SessionFactory;

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

**import** org.springframework.stereotype.Repository;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.hexa.entity.Account;

@Repository("mydao")

**public** **class** AccountDaoImpl **implements** AccountDao{

@Autowired

**private** SessionFactory sfac;

@Override

**public** **int** addAccount(Account account) {

Session sess = sfac.getCurrentSession(); // If called directly will not happen as the transaction will not be created

sess.save(account);

**return** 1;

}

@Override

**public** **int** updateAccount(Account account) {

Session sess = sfac.getCurrentSession();

sess.update(account);

**return** 1;

}

@Transactional(readOnly = **true**) // When transaction is not required for the method , annotation is optional

@Override

**public** Account getAccount(**int** accountId) {

Session sess = sfac.openSession();

Account acc = sess.get(Account.**class**, accountId);

**return** acc;

}

}

Iser.java

**package** com.hexa.service;

**import** java.util.List;

**import** com.hexa.entity.Account;

**public** **interface** ISer {

**int** addAccounts(List<Account> lst);

**int** transferFund(**int** from, **int** to, **double** amt);

}

**SerImpl.class**

**package** com.hexa.service;

**import** java.util.List;

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

**import** org.springframework.stereotype.Service;

**import** org.springframework.transaction.annotation.Transactional;

**import** com.hexa.dao.AccountDao;

**import** com.hexa.entity.Account;

@Service("myser")

**public** **class** SerImpl **implements** ISer{

@Autowired

**private** AccountDao dao;

@Transactional(readOnly = **false**)

@Override

**public** **int** addAccounts(List<Account> lst) {

**for**(Account acc : lst) {

dao.addAccount(acc);

}

**return** 0;

}

@Override

**public** **int** transferFund(**int** from, **int** to, **double** amt) {

Account facc = dao.getAccount(from);

Account tacc = dao.getAccount(to);

**if**(facc.getBal() < amt) {

**return** 0;

}

facc.setBal(facc.getBal() - amt);

facc.setBal(facc.getBal() - amt);

dao.updateAccount(facc);

dao.updateAccount(tacc);

**return** 1;

}

}

output:

Hibernate:

insert

into

account

(bal, cname, aid)

values

(?, ?, ?)

Hibernate:

insert

into

account

(bal, cname, aid)

values

(?, ?, ?)

**Spring MVC**

* Add Tomacat to the server
* Create dynamic web project
* Refer yur tomcat library by
* Right click => Properties => Build Path => Add Library => Server Runtime => Choose yur Tomcat
* Copy all the jar provided by trainer in lib folder under web-inf FOLDER
* COPY THE WEB.XML AND THE SpringMvcServlet.xml under WEB-INF folder
* Make the following change in springMvc-servlet.xml
* class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>
* <property name=*"prefix"* value=*"/WB\_INF"* />
* Create package com.hexa.web for controller and create a class StudentController
* Create a jsp file under web-inf folder
* Change charset to UTF-8 and remove DOCTYPE keep only <!DOCTYPE html>

**package** com.hexa.web;

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.web.bind.annotation.RequestMapping;

@Controller // This class has to be maintained by IOC

**public** **class** Studentcontroller {

@RequestMapping("/students")

**public** String displayStudents(Model model) {

model.addAttribute("title","Student details");

**return** "StudentView"; //return the view name

}

}

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

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

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

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

*http://www.springframework.org/schema/beans/spring-beans-3.0.xsd*

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

*http://www.springframework.org/schema/context/spring-context-3.0.xsd*

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

*http://www.springframework.org/schema/tx/spring-tx-3.0.xsd*

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

*http://www.springframework.org/schema/aop/spring-aop-3.0.xsd*

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

*http://www.springframework.org/schema/mvc/spring-mvc-3.0.xsd"*>

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

<bean id=*"viewResolver"*

class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*>

<property name=*"prefix"* value=*"/WEB-INF"* />

<property name=*"suffix"* value=*".jsp"* />

</bean>

</beans>

<!-- </bean>

<bean id="messageSource"

class="org.springframework.context.support.ResourceBundleMessageSource">

<property name="basename" value="global"/>

</bean> -->

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

<web-app xmlns:xsi=*"http://www.w3.org/2001/XMLSchema-instance"* xmlns=*"http://java.sun.com/xml/ns/javaee"* xsi:schemaLocation=*"http://java.sun.com/xml/ns/javaee http://java.sun.com/xml/ns/javaee/web-app\_3\_0.xsd"* id=*"WebApp\_ID"* version=*"3.0"*>

<display-name>SpringMvc</display-name>

<welcome-file-list>

<welcome-file>Home.jsp</welcome-file>

</welcome-file-list>

<servlet>

<servlet-name>SpringMvc</servlet-name>

<servlet-class>org.springframework.web.servlet.DispatcherServlet</servlet-class>

<load-on-startup>1</load-on-startup> <!-- EAGER iNITIALIZATION -->

</servlet>

<servlet-mapping>

<servlet-name>SpringMvc</servlet-name>

<url-pattern>\*.htm</url-pattern>

</servlet-mapping>

<!-- <listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/spring-jdbc.xml</param-value>

</context-param> -->

</web-app>

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<!DOCTYPE html">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<h1 align=*"center"*> ${title} </h1>

<hr size=*"2"* style="color:*red*"/>

</body>

</html>

* Add the project in tomcat and start the server
* Run the code
* Now JDBC
  + Copy the spring-jdbc xml from previous project
  + Copy the dao and entity packages from the previous project
  + Add the below lines to the web.xml file

<listener>

<listener-class>org.springframework.web.context.ContextLoaderListener</listener-class>

</listener>

<context-param>

<param-name>contextConfigLocation</param-name>

<param-value>/WEB-INF/spring-jdbc.xml</param-value>

</context-param>

Modify StudentController and StudentView files

**package** com.hexa.web;

**import** java.util.List;

**import** javax.servlet.http.HttpServletRequest;

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

**import** org.springframework.stereotype.Controller;

**import** org.springframework.ui.Model;

**import** org.springframework.web.bind.annotation.RequestMapping;

**import** org.springframework.web.bind.annotation.RequestParam;

**import** com.hexa.dao.IDao;

**import** com.hexa.entity.Student;

@Controller // This class has to be maintained by IOC

**public** **class** Studentcontroller {

@Autowired

**private** IDao dao;

@RequestMapping("/students")

**public** String displayStudents(@RequestParam("dname") String dept, Model model) {

List<Student> lst = dao.getStudents(dept);

model.addAttribute("title", "Student details");

model.addAttribute("slist",lst);

**return** "StudentView"; // returning the view name

}

}

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<%@taglib uri=*"http://java.sun.com/jsp/jstl/core"* prefix=*"c"*%>

<!DOCTYPE html">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<h1 align=*"center"*>${title}</h1>

<hr size=*"2"* style="color: *red*" />

<table cellpadding=*"10"* bgcolor=*"wheat"* style="margin: *0 auto*">

<tr>

<td>Student Id</td>

<td>Name</td>

<td>DOB</td>

<td>Department</td>

<td>Marks</td>

</tr>

<c:forEach items=*"*${slist}*"* var=*"s"*>

<tr>

<td>${s.sId}</td>

<td>${s.sName}</td>

<td>${s.dob}</td>

<td>${s.dept}</td>

<td>${s.marks}</td>

</tr>

</c:forEach>

</table>

</body>

</html>

**INTERNATIONALIZATION**

Create a file named messages.properties in src folder Add the following data

The key and value can be anything, not according to entity class

stuid=Student Id

stuname=Student Name

marks=Student Marks

dept=Department

dob=Date Of Birth

create another file messages\_de.properties in src folder. All properties must have same key value. Add the following data

stuid=Studenten ID

stuname=Name des Studenten

marks=Schüler Noten

dept=Abteilung

dob=Geburtsdatum

messages\_hi.properties

stuid=\u091B\u093E\u0924\u094D\u0930 \u0906\u0908\u0921\u0940

stuname=\u091B\u093E\u0924\u094D\u0930 \u0915\u093E \u0928\u093E\u092E

marks=\u0935\u093F\u0926\u094D\u092F\u093E\u0930\u094D\u0925\u0940 \u0905\u0902\u0915

dept=\u0935\u093F\u092D\u093E\u0917

dob=\u091C\u0928\u094D\u092E \u0915\u0940 \u0924\u093E\u0930\u0940\u0916

messages\_ta\_IN.properties

stuid=\u0BAE\u0BBE\u0BA3\u0BB5\u0BB0\u0BCD \u0B85\u0B9F\u0BC8\u0BAF\u0BBE\u0BB3\u0BAE\u0BCD

stuname=\u0BAE\u0BBE\u0BA3\u0BB5\u0BA9\u0BCD \u0BAA\u0BC6\u0BAF\u0BB0\u0BCD

marks=\u0BAE\u0BBE\u0BA3\u0BB5\u0BB0\u0BCD \u0BAE\u0BA4\u0BBF\u0BAA\u0BCD\u0BAA\u0BC6\u0BA3\u0BCD

dept=\u0BA4\u0BC1\u0BB1\u0BC8

dob=\u0BAA\u0BBF\u0BB1\u0BA8\u0BCD\u0BA4 \u0BA4\u0BC7\u0BA4\u0BBF

Add the below code in springmvc servlet .xml file <bean id=*"messageSource"*

class=*"org.springframework.context.support.ResourceBundleMessageSource"*>

<property name=*"messages"* value=*"global"*/>

</bean>

**Modify StudentView**

<%@ page language=*"java"* contentType=*"text/html; charset=UTF-8"*

pageEncoding=*"UTF-8"*%>

<%@taglib uri=*"http://java.sun.com/jsp/jstl/core"* prefix=*"c"*%>

<%@taglib uri=*"http://www.springframework.org/tags"* prefix=*"spring"*%>

<!DOCTYPE html">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=UTF-8"*>

<title>Insert title here</title>

</head>

<body>

<h1 align=*"center"*>${title}</h1>

<hr size=*"2"* style="color: *red*" />

<table cellpadding=*"10"* bgcolor=*"wheat"* style="margin: *0 auto*">

<tr>

<tr>

<td><spring:message code=*"stuid"*></spring:message></td>

<td><spring:message code=*"stuname"*></spring:message></td>

<td><spring:message code=*"dob"*></spring:message></td>

<td><spring:message code=*"dept"*></spring:message></td>

<td><spring:message code=*"marks"*></spring:message></td>

</tr>

</tr>

<c:forEach items=*"*${slist}*"* var=*"s"*>

<tr>

<td>${s.sId}</td>

<td>${s.sName}</td>

<td>${s.dob}</td>

<td>${s.dept}</td>

<td>${s.marks}</td>

</tr>

</c:forEach>

</table>

</body>

</html>

**Standalone spring Boot Application**

* File -> New -> Spring starter Project
* Give a name : SpringJdbcBoot
* Packaging : jar (for standalone)
* War : (For web application)
* Packege Name : com.hexa
* Click on Next
* Boot version : 1.5.10
* Select JDBC and MYSQL
* Click Next -> Next -> Finish

Once the project is created :

* In pom check if the selected dependencies are available
* From SpringJdbcAnnoattion copy and paste the dao and entity package in src/main/java
* Base package name must be com.hexa
* Spring boot starter jdbc will create jdbc template hence it is not require to provide it separately in xml or the config file.
* In DaoImpl.java file add
  + @Autowired
  + Private
* In src/main/resources -> application.properties add the following line
  + spring.datasource.driver-class-name=com.mysql.jdbc.Driver
  + spring.datasource.url=jdbc:mysql://localhost:3306/test
  + spring.datasource.username=root
  + spring.datasource.password=root
* Spring boot can be used to develop web as well as standalone application
* Go to SpringJdbcApplication.java. Add

Implements CommandLineRunner

@autowired

Private IDao dao

@override

Public void run(String … arg0) throws Exception {

List<Student> lst = dao.getStudents(“ECE”);

Lst.foreach(System.out::println);

}

* Execute the code Run As- Java Application