**LOCATION CODDING**

**Project Setups:--**

**Software Required :---**

1>JDK 1.7 (7.0) 32-bit)

2>Tomcat 7.0 (32-bit).

3>Eclipse Luna/Juno/Kepler (32-bit)

4>Oracle 10G XE or MySQL 5 Server

5>Project jars.

**Steps#1:--** Open Eclipse and change perspective to Java EE.

=>Window ===>open perspective===>Other===>Java EE===>Finish.

**Step#2:--** Add JDK/JRE system library to project.

=>Window ===>Preferences ===>search with “installed JRE” & select same===>Click on Add button

=>select “standard VM “===>next===>Browse (C:\Program Files\Java\jdk1.8.0\_171)===>finish.

**Steps#3:--**Make it as default JRE

=>click on box ===>Finish.

**Step#4:--** Setup Tomcat server in Eclipse.

=>Window ===>show view===>server===>click on server view===>right click===>server===>Add (new server)===>

Choose “Apache Tomcat 7===>next ===>Browse for Location (C:\Program Files\Apache Software Foundation\Tomcat 7.0\_Tomcat7.84\bin)==>finish.

**Steps#5:--** Create one Dynamic web project

=>File ===>new dynamic web project ===>enter project name (VendorApp)===>select===>select

1>Target Runtime :Apache Tomcat 7.0

2>Dynamic web module version :2.5===>Next===>next===>

choose click box (“[V] Generate Deployment Descriptor…”=>Finish

**Steps#5:--** Downlaod jars and Extract to folder

=>Select all jars and copy (ctrl+C)===>goto eclipse project===>Right click on “lib” folder and past.

========================================================================================================

=>To develop one project we should use Layers Design (3+1) those are:

1>Presentation Layer (PL).

2>Service Layer (SL).

3>Data Access Layer (DAL).

4>Integration Layer (IL).

=>In these layers, we should write 10 java files given ordered list as.

1.>Model class (C)

2>IDao (POJI) in DAL (I)

3>DaoImpl (POJO) in DAL (C)

4>IService (POJI) in DAL (I)

5>IServiceImpl (POJO) in SL (C)

6>Controller in PL (C)

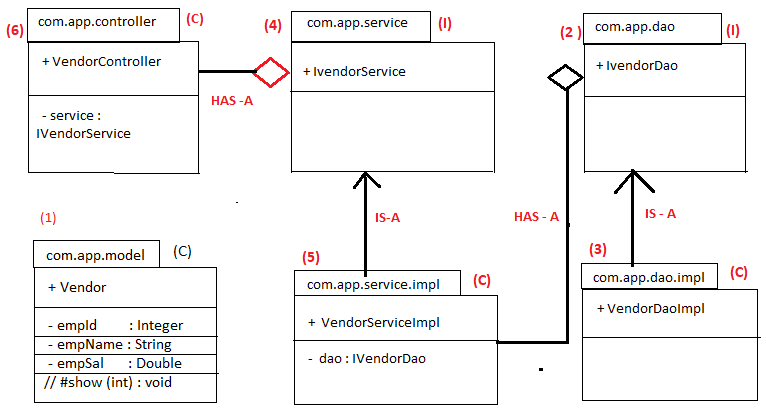
7>Util //Optinal (C)

8>Validator //Optional (C)

9>View (C)

10>ServiceProvider or ServiceConsumer in IL (C)

**Module Basic Layer Design:--**



**Types of files /Coding in Projects:--**

**1>One time Coding :--** These files are written in project only one time which work for all modules (complete project).

=>This kind of code in also called as setup code.

=>All XMl configuration comes under this one time setup/coding.

**2>Module wise coding :--**

=>These files are written for every module.

Ex> If you have 10 modules in your project then these files must we written 10 times.

|  |  |
| --- | --- |
| **One time coding** | **Module –wise coding** |
| 1. Web.xml. 2. Spring config [DBMS, ASFB, HT,   ViewResource, Activate Annotations]   1. Spring view xml | 1>Model class  2>DAL : POJI-POJO  3>SL : POJI-POJO  4>Util, Validator  5>PL : Controller  6>UI Pages (JSPs)  7>JS, CSSCC, Views, Aspect |

**1>ONE TIME CODDING:---**

**1>web.xml (Front Controller):--**

=>In Spring Web MVC application, “Dispatcher Servler” behaves like entry and exit gate to application.

=>It takes request and provides finally response to client.

=>It must be configured in “web.xml” file using “DirectoryMatch URL Pattern”.

=>URL Patterns are 3 types in web.xml, those are.

a>Exact Match. ex:-- /abc, /xyz

b>Extension Match (.) Ex : \*.do, \*.ab,…..

c>Directory Match (/) Ex:-- /ab/\*, /mvc/\*

**1>web.xml Code (Under WEB-INF folder):--**

<?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\_2\_5.xsd"*

id=*"WebApp\_ID"* version=*"2.5"*>

<context-param>

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

<param-value>/WEB-INF/sample-servlet.xml</param-value>

</context-param>

<listener>

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

</listener>

<servlet>

<servlet-name>sample</servlet-name>

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

</servlet>

<servlet-mapping>

<servlet-name>sample</servlet-name>

<url-pattern>/mvc/\*</url-pattern>

</servlet-mapping>

<servlet>

</web-app>

**Hibernate Templet:--**

=>This is a predefined class given by spring ORM (Hibernate) which is used to perform DB operations like save, update, delete, and select in less lines of code using Hibernate API internally.

**1> ht.save(obj);**  :-- It will convert model class object (DTO=Data transfer object) to DB table row.

**2>ht.update (obj);** :--It will update DB table row, by taking DTO as input Based on primary key other values/columns are updated. If primary key not found does nothing.

**3>ht.delet(obj);**

=> It will take one Object having only primary key value and based on primary key (if found) row will be deleted.

**4>ht.get (T.class, id) :-- T<Object>**

=>It will select one row based on Primary key value. This Row will be converted to one Model class object also called as DTO.

**5>ht.loadAll(T.class); List<T>**

=>To select all rows in a DB table use loadAll method, which convert all rows to list of objects.

**2>Create a XML file (sample-servlet.xml) under WEB-INF folder:--**

**Spring Config file one-time-setup code :--**

=>This code must be written only one time for all modules.

=>Here coding is written in this file for.

a>Activation of Annotations.

b>ViewResolver (prefix/suffix).

c>DriverManagerDataSource.

d>AnnotationSessionFactoryBean.

e>HibernateTemplet.

**Data Access Layer:--** DAL will connect to DB to perform operation like, update, delete and select.

=>Here, these must be configured in application using XML config (or java config) one time in bellow order.

1>DriverManagerDataSource.

2>AnnotationSessionFactoryBean.

3>HibernateTemplet.

**Hibernate Properties :--** Properties means storing data in key=value format, both are of type String.

=>To do hibernate programming, use bellow keys given with example values.

**i>dialect :--** It is a pre defined class in hibernate. It generates SQL query when you perform any operation.

=>Database to database Dialect value changes. Like

Ex1:-- hibernate.dialect =org.hibernate.dialect.MySQLDialect

Ex2:-- hibernate.dialect =org.hibernate.dialect.OracleDialect

**ii>show\_sql (default value is false):--** It will display SQL query generated by Dialect on console if value is “true”.

Ex:-- hibernate.show\_sql = true

**iii>format\_sql (Default value is false):--** It will display SQL query step by step on console.

Ex:-- hibernate.format\_sql = true;

**iv>hbm2ddl.auto (Default value is validate):--** It is used to create tables.

=>Possible values are : validate, create, update and create-drop.

a>validate :--Hibernate create no table, Programmer has to take care of tables.

b>create :-- This option create table every time (new table), if table exist it will be dropped.

c>update :-- Checks for table if exist use same else create new table.

d>create-drop :-- This option create new table and drops after operation done.

Ex:-- hibernate.hbm2ddl.auto = create

=>Here only dialect is required, other keys are optional (default value is taken if key is not provided).

**Sample-servlet.xml:--**

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

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

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

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

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

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

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

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-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/task*

*http://www.springframework.org/schema/task/spring-task-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/aop/spring-aop-3.0.xsd) *"*>

<!-- Activation of annotations -->

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

<aop:aspectj-autoproxy/>

<!-- ViewResolver Config -->

<bean class=*"org.springframework.web.servlet.view.InternalResourceViewResolver"*

p:prefix=*"/WEB-INF/views/"* p:suffix=*".jsp"* />

<!-- DMDS (DriverManagerDataSource) Object config -->

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

name=*"dsObj"* p:driverClassName=*"oracle.jdbc.driver.OracleDriver"*

p:url=*"jdbc:oracle:thin:@localhost:1521:xe"* p:username=*"system"*

p:password=*"system"* />

<!-- ASFB (AnnotationSeesionFactoryBean) Object config -->

<bean class=*"org.springframework.orm.hibernate3.annotation.AnnotationSessionFactoryBean"*

name=*"sfObj"*>

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

<property name=*"hibernateProperties"*>

<props>

<prop key=*"hibernate.dialect"*>org.hibernate.dialect.OracleDialect</prop>

<prop key=*"hibernate.format\_sql"*></prop>

<prop key=*"hibernate.show\_sql"*>true</prop>

<prop key=*"hibernate.hbm2ddl.auto"*>update</prop>

</props>

</property>

<property name=*"annotatedClasses"*>

<list>

<value>com.app.model.Location</value>

<value>com.app.model.Document</value

<!-- Add all modules fully qualify name one by one down -->

</list>

</property>

</bean>

<!-- HT (HibernateTemplet object config -->

<bean class=*"org.springframework.orm.hibernate3.HibernateTemplate"* name=*"htObj"*>

<property name=*"sessionFactory"* ref=*"sfObj"* />

</bean>

<bean class=*"org.springframework.web.multipart.commons.CommonsMultipartResolver"*

name=*"multipartResolver"*>

<property name=*"maxUploadSize"* value=*"20971520"* />

<property name=*"maxInMemorySize"* value=*"10485760"* />

</bean>

</beans>

**3>Create a Model class (Location):--**

=>Here model means data. Data will be transferred from UI to DB and DB to UI in Object format also called as DTO (Data Transfer Object) or VO (Value Object).

=>By using Model class these DTO or VO Objects are created by frameworks (Spring and Hibernate) in Simple container.

=>This Model class can also be called as Entity class/POJO class.

=>Here @Entity and @Id are required. Here @Id indicates Primary Key. Every table must have one primary key.

=> @Table and @Column are optional, If not mention then by default class name is taken as Table name and variable name as column name is the.

=>Model class must be mapped with UI form also.

---> Here No. Of variable in model class = No of <inputs> in UI form.

=>Every Input name (<input name = “ “>) must match with variable name.

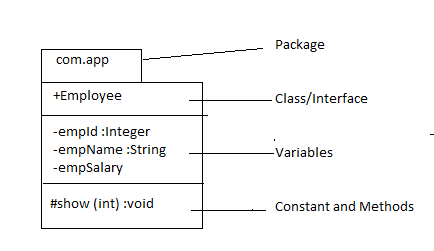
1>UML (Unified Modeling Language):- It is used to represent Design format. Application for Java/.net/Php etc..

=>Here below mention public (+), private (-), protected (#) and default (no symbol).

2>Java Relations:--

i>IS-A (Inheritance) extends/Implements.

ii>HAS-A Association (Use child as dataType in parent class, create one variable). Here Rhombus ( ) indicates child class.



**package** com.app.model;

@Entity //import javax.persistence.Entity;

@Table(name = "loc\_tab") //import javax.persistence.Table;

**public** **class** Location

{

@Id //javax.persistence.Id;

@Column(name ="loc\_id") //javax.persistence.Column;

@GeneratedValue(generator ="loc\_gen") //javax.persistence.GeneratedValue;

@GenericGenerator(name ="loc\_gen", strategy ="increment")

//org.hibernate.annotations.GenericGenerator;

**private** Long locId;

@Column(name ="c\_date")

@DateTimeFormat(pattern = “yyyy-MM-dd hh:mm:ss”) //org.springframework.format.annotation.DateTimeFormat;

@Temporal(TemporalType.***TIMESTAMP***)

//javax.persistence.Temporal; ,javax.persistence.TemporalType;

@CreatedDate //org.springframework.data.annotation.CreatedDate;

**private** Date createdOn; //java.util.Date;

@Column(name ="m\_date")

@DateTimeFormat(pattern = “yyyy-MM-dd hh:mm:ss”) //org.springframework.format.annotation.DateTimeFormat;

@Temporal(TemporalType.***TIMESTAMP***)

@LastModifiedDate //org.springframework.data.annotation.LastModifiedDate;

**private** Date modifiedOn;

@Column(name="l\_name")

**private** String locName;

@Column(name="l\_type")

**private** String locType;

//Order must be followed

//1. Alt+Shift+S,O (De-select all) (Generate default Constructor)

//2. Alt+shift+S,O (Deselect all, select P.k) (Generate ID based parameterized constructor)

//3. Alt+shift+S,O (Deselect ID(P.k), CreatedDate, ModifiedDate) (Generate parameterized constructor)

//4. Alt+shift+S, R (Generate setter and Getter)

//5. Alt+shift+S, S (Generate toString String Method).

}

**3>Create one Interface “ILocationDao”:--**

**package** com.app.dao;

**public** **interface** ILocationDao

{

/\*1>\*\*\*\*\*Save Operation \*\*\*\*\*\*\*\*\*\*/

**public** **int** saveLocation(Location loc);

/\*2>\*\*\*\*\*\*\*\*Get All records\*\*\*\*\*\*\*\*/

**public** List<Location> getAllLocations();

/\*3>\*\*\*\*\*\*Delete specific records by Id based\*\*\*\*\*/

**public** **void** deleteLocById(**int** locId);

/\*4>\*\*\*\*\*\*\*\*get Specific data by Id based\*\*\*\*\*\*\*/

**public** Location getLocationById(**int** locId);

/\*5>\*\*\*\*\*\*\*\*update data (record) based on Id\*\*\*\*\*\*\*\*/

**public** **void** updateLocation(Location loc);

/\*6>\*\*\*get number of Count based on specific column\*\*\*\*\*/

**public** List<Object[]> getLocTypeWiseCount();

}

**4>Create one Class “LocationDaoImpl” and implements “ILocationDao”:--** implements IDao abstract method in DaoImpl using ht (dependency).

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

@Repository //org.springframework.stereotype.Repository;

**public** **class** LocationDaoImpl **implements** ILocationDao

{

@Autowired //org.springframework.beans.factory.annotation.Autowired;

**private** HibernateTemplate ht; //org.springframework.orm.hibernate3.HibernateTemplate;

/\*1>\*\*\*\*\*Save Operation \*\*\*\*\*\*\*\*\*\*/

@Override

**public** **int** saveLocation(Location loc) {

**return** (Integer)ht.save(loc);

}

/\*2>\*\*\*\*\*\*\*\*Get All records\*\*\*\*\*\*\*\*/

@Override

**public** List<Location> getAllLocations() {

**return** ht.loadAll(Location.**class**);

}

/\*3>\*\*\*\*\*\*Delete specific records by Id based\*\*\*\*\*/

@Override

**public** **void** deleteLocById(**int** locId) {

ht.delete(**new** Location(locId));

}

/\*4>\*\*\*\*\*\*\*\*get Specific data by Id based(Edit)\*\*\*\*\*\*\*/

@Override

**public** Location getLocationById(**int** locId) {

**return** ht.get(Location.**class**, locId);

}

/\*5>\*\*\*\*\*\*\*\*update data (record) based on Id\*\*\*\*\*\*\*\*/

@Override

**public** **void** updateLocation(Location loc) {

ht.update(loc);

}

/\*6>\*\*\*get number of Count based on specific column\*\*\*\*\*/

@Override

**public** List<Object[]> getLocTypeWiseCount() {

String hql="select locType,count(locType) "

+" from "+Location.**class**.getName()

+" group by locType ";

List<Object[]> lst=ht.find(hql);

**return** lst;

}

}

**4>Create one Interface “ILocationService” :--**

**package** com.app.service;

**public** **interface** ILocationService

{

/\*1>\*\*\*\*\*Save Operation \*\*\*\*\*\*\*\*\*\*/

**public** **int** saveLocation(Location loc);

/\*2>\*\*\*\*\*\*\*\*Get All records\*\*\*\*\*\*\*\*/

**public** List<Location> getAllLocations();

/\*3>\*\*\*\*\*\*Delete specific records by Id based\*\*\*\*\*/

**public** **void** deleteLocById(**int** locId);

/\*4>\*\*\*\*\*\*\*\*get Specific data by Id based\*\*\*\*\*\*\*/

**public** Location getLocationById(**int** locId);

/\*5>\*\*\*\*\*\*\*\*update data (record) based on Id\*\*\*\*\*\*\*\*/

**public** **void** updateLocation(Location loc);

/\*6>\*\*\*get number of Count based on specific column\*\*\*\*\*/

**public** List<Object[]> getLocTypeWiseCount();

}

**5>Create one “LocationServiceImpl” class and implements “ILocationService” Interface:--**

=>implement abstract method in ServiceImpl using dao (IDao), call dao method here (same name).

**package** com.app.service.impl;

@Service //org.springframework.stereotype.Service;

**public** **class** LocationServiceImpl **implements** ILocationService

{

@Autowired

**private** ILocationDao dao;

/\*1>\*\*\*\*\*Save Operation \*\*\*\*\*\*\*\*\*\*/

@Override

**public** **int** saveLocation(Location loc) {

**return** dao.saveLocation(loc);

}

/\*2>\*\*\*\*\*\*\*\*Get All records\*\*\*\*\*\*\*\*/

@Override

**public** List<Location> getAllLocations() {

List<Location> locList=dao.getAllLocations();

**if**(locList!=**null** && locList.size()>1){

Collections.*sort*(locList);

}

**return** locList;

}

/\*3>\*\*\*\*\*\*Delete specific records by Id based\*\*\*\*\*/

@Override

**public** **void** deleteLocById(**int** locId) {

dao.deleteLocById(locId);

}

/\*4>\*\*\*\*\*\*\*\*get(Edit) Specific data by Id based\*\*\*\*\*\*\*/

@Override

**public** Location getLocationById(**int** locId) {

**return** dao.getLocationById(locId);

}

/\*5>\*\*\*\*\*\*\*\*update data (record) based on Id\*\*\*\*\*\*\*\*/

@Override

**public** **void** updateLocation(Location loc) {

dao.updateLocation(loc);

}

/\*6>\*\*\*get number of Count based on specific column\*\*\*\*\*/

@Override

**public** List<Object[]> getLocTypeWiseCount() {

**return** dao.getLocTypeWiseCount();

}

}

**6>Create one “UomController” class:--**

=>To handle request made by client define one Controller class in application having multiple methods (request method).

=>Every request methods must be linked with one Unique URL.

**1>ModelAttribute:--** It is an Object created by Spring container, on submitting from data. When end user, enters data in hTML form and clicks on submit button.

**Spring Container :--**

a>Creates object to model class.

b>Reads data from HTML form.

c>Parse data if required.

d>Set data to ModelAttribute

NOTE:--Object name will be class name and first letter lower case (camel case format).

Ex:-- @ModelAttribute (“objectname”) ModelClassName LocalVariableName;

**2>ModelMap :--** It isused to send data from controller (PL)to view (UI) in key =value format, key must be String type, value can be primitive, Object or Collection.

=>use method addAttibute (K, V). It will add data to model map.

=>Use same key (case-sensetive) at UI to read data from model map using EL (Expression Language).

Ex:-- ${message}, ${empObj}….. etc.

**package** com.uday.app.controller;

@Controller //org.springframework.stereotype.Controller;

@RequestMapping("/uom") //org.springframework.web.bind.annotation.RequestMapping;

**public** **class** UomController

{

@Autowired //org.springframework.beans.factory.annotation.Autowired;

**private** IUomService service;

//To show Register Page

@GetMapping("/show") //org.springframework.web.bind.annotation.GetMapping;

**public** String showRegister(ModelMap map)

{

//Send ModelAttribute Object to form

map.addAttribute("uom", **new** Uom());

//Get ListData(DropDown, CheckBox, RadioButtons) from UI Screen

map.addAttribute("uomTypes", util.getUomTypes());

//Specify UI PageName

**return** "UomRegister"; }

//Save data into DB without validation.

@PostMapping("/register") //org.springframework.web.bind.annotation.PostMapping;

**public** String saveData(@ModelAttribute Uom uom, ModelMap map)

{

//Save data into DB using service layer

Long uomId =service.save(uom);

//Clear form value (Model Object) after save

map.addAttribute("uom", **new** Uom());

//send message to UI after save

map.addAttribute("message", "Uom Created with Id : "+uomId);

**return** "UomRegister";

}

**7> Create one “UomRegister.jsp” page (src/main/webapp/WEB-INF/views/UomRegister.jsp):--**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

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

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

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Insert title here</title>

</head>

<body>

<h1>Welcome to Uom Register</h1>

<form:form action=*"register"* method=*"post"* modelAttribute=*"uom"*>

<pre>

Uom Type: <form:select path=*"uomType"*>

<form:option value=*""*> ---Select---</form:option>

<form:option value=*"PACKING"*>PACKING</form:option>

<form:option value=*"NON-PACKING"*>NON-PACKING</form:option>

<form:option value=*"NA"*>NA</form:option>

</form:select>

Uom Model: <form:input path=*"uomModel"* />

Desc: <form:textarea path=*"description"* />

<input type=*"submit"* value=*"Create Uom"* />

</pre>

</form:form>

<b>${message}</b>

</body>

</html>

**2> Validation:---**

**1>Create one “messages.properties” file inside src/main/resource:--**

uom.reg.type.error = Select one UOM Type

uom.reg.model.error = Enter UOM Model

uom.reg.desc.error = Enter Description

**2>Create one “UomValidator” class and implements “Validator” Interface:--**

**package** com.uday.app.validator;

@Component //org.springframework.stereotype.Component;

**public** **class** UomValidator **implements** Validator //org.springframework.validation.Validator;

{

@Override

**public** **boolean** supports (Class<?> clazz)

{

**return** Uom.**class**.equals(clazz);

}

@Override

**public** **void** validate (Object target, Errors errors) //org.springframework.validation.Errors;

{

//Normal Validator

//ValidationUtils.rejectIfEmptyOrWhitespace(errors, "uomType", "uom.reg.type.error",

"Enter defaultMessage"); //Direct supply message inside body

ValidationUtils.*rejectIfEmptyOrWhitespace*(errors, "uomType", "uom.reg.type.error");

ValidationUtils.*rejectIfEmptyOrWhitespace*(errors, "uomModel", "uom.reg.model.error");

ValidationUtils.*rejectIfEmptyOrWhitespace*(errors, "description", "uom.reg.desc.error");

}

}

**3>UomController code to be added/modified (with validation):--**

**package** com.uday.app.controller;

//add UomValidator (HAS-A) inside Controller class

@Autowired

**private** UomValidator validator;

//Re write save method (2nd method)

@PostMapping("/register")

**public** String saveData(@ModelAttribute Uom uom, Errors errors, ModelMap map)

{

/\*\*\* Before Save to validation\*\*\*/

validator.validate(uom, errors);

**if**(!errors.hasErrors())

{

Long uomId =service.save(uom);

map.addAttribute("uom", **new** Uom());

map.addAttribute("message", "Uom Created with Id :"+uomId);

}

**return** "UomRegister";

}

**4>In “UomRegister.jsp” display errors using:--**

<form:errors path = “uomType” />

<form:errors path = “uomModel” />

<form:errors path = “description” />

**3>Fetching all Rows from DB table:--**

**1>Add method in “IUomService”:--**

public List<Uom> getAll();

**2>Implement in “UomServiceImpl” as:--**

**package** com.uday.app.service.impl;

**public** List<Uom> getAll()

{

List<Uom> uomList =repo.findAll();

Collections.*sort*(uomList, **new** Comparator<Uom>()

{

**public** **int** compare (Uom O1, Uom O2)

{

**return** O1.getUomModel().compareToIgnoreCase(O2.getUomModel());

}

});

**return** uomList;

}

**3>Add Bellow method in “UomController” class:--**

**package** com.uday.app.controller;

@GetMapping("/all") //Show All Uom Records

**public** String getAll(ModelMap map)

{

//map.addAttribute("uomList", service.getAll());

map.addAttribute("uomList", service.getAll());

**return** "UomData";

}

**4>Create “UomData.jsp” in views folder:---**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"* pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 ransitional//EN"http://www.w3.org/TR/html4/loose.dtd">

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

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

<html>

<body><h1>Welcome to UomData</h1>

<table border =*"1"*>

<tr>

<th>ID</th> <th>TYPE</th>

<th>MODEL</th> <th>DESC</th>

<th>CREATED</th> <th>MODIFIED</th>

</tr>

<form:form action=*"all"* method=*"get"* modelAttribute=*"uom"*/>

<c:forEach items =*"*${uomList}*"* var =*"uom"*>

<tr>

<td><c:out value=*"*${uom.uomId}*"* /></td>

<td><c:out value=*"*${uom.uomType}*"* /></td>

<td><c:out value=*"*${uom.uomModel}*"* /></td>

<td><c:out value=*"*${uom.description}*"* /></td>

<td><c:out value=*"*${uom.createdOn}*"* /></td>

<td><c:out value=*"*${uom.modifiedOn}*"* /></td>

<td> <a href =*"Edit?uomId=*${uom.uomId}*"*>Edit</a></td>

<td> <a href =*"Delete?uomId=*${uom.uomId}*"*>Delete</a></td>

</c:forEach>

</table>

</body>

</html>

**4>Delete Specific Record by using uomId:---**

**1>Create one Hyperlink using static path and Dynamic path (URLReWriting):---**

=>Add bellow code in “UomData.jsp” page.

Ex:-- <a href =”delete?uomId = ${uom.uomId}”>DELETE</a>

<form:form action=*"all"* method=*"get"* modelAttribute=*"uom"*/>

<c:forEach items =*"*${uomList}*"* var =*"uom"*>

<tr>

<td><c:out value=*"*${uom.uomId}*"* /></td>

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

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

<td><c:out value=*"*${uom.modifiedOn}*"* /></td>

<td> <a href =*"delete?uomId=*${uom.uomId}*"*>Delete</a></td>

</c:forEach>

**2>Add bellow method in “IUomService” Intreface: ---**

public void delete(Long uomId);

**3>Add method in “UomServiceImpl” class: ---**

**public** **void** delete(Long uomId) {

repo.delete(uomId);

}

**4>Add method in “Controller” class: ---**

//To Delete specific record by uomId

@GetMapping("delete")

**public** String delet(@RequestParam Long uomId) //Long Must be Model class variable name

{

service.delete(uomId);

**return** "redirect:all";

}

**5>To Edit specific Data in Database: ---**

**1>Create one Hyperlink using static path and Dynamic path (URLReWriting):---**

<a href =”edit?uomId = ${uom.uomId}”>EDIT</a>

**2>Add bellow method in “IUomService” Intreface: ---**

public void update(Uom uom); //update data based on ID(pk)

public void getOne(Long uomId); //update data based on ID(pk)

**3>Add method in “UomServiceImpl” class: ---**

//To Delete Specific Record

**public** **void** update(Uom uom)

{

repo.save(uom);

}

//To Edit Specific Record

**public** Uom getOne(Long uomId)

{

Uom uom =repo.getOne(uomId);

**return** uom;

}

**4>Add two methods in “Controller”cx class: ---**

//Edit Specific record by uomId

@GetMapping("/edit")

**public** String edit(@RequestParam Long uomId, ModelMap map)

{

Uom uom = service.getOne(uomId);

map.addAttribute("uom", uom);

**return** "UomDataEdit";

}

//Update Specific record by uomId

@PostMapping("/update")

**public** String update(@ModelAttribute Uom uom)

{

service.update(uom);;

**return** "redirect:all";

}

6>Dynamic UI Construction using spring form tags: ---

**1> Create one “UomUtil” class (and apply @Component at class level):---**

**package** com.uday.app.util;

@Component //org.springframework.stereotype.Component;

**public** **class** UomUtil

{

**public** List<String> getUomTypes()

{

//1 method using Enum

**return** Arrays.*asList*("PACKING","NO-PACKING","NA");

//2 Method Using List

/\*List<String> list = new ArrayList<String>();

{

list.add("PACKING");

list.add("No-PACKING");

list.add("NA");

return list;

}\*/

}

//Add all above List<String> Field name

**public** **void** addUiComponents(ModelMap map) //org.springframework.ui.ModelMap;

{

map.addAttribute("uomTypes", getUomTypes());

}

}

**2>In “UomController” class: ---**

@Autowired //Make HAS-A relation with “UomUtil” class

**private** UomUtil util; //com.uday.app.util.UomUtil;

/\*In showRegister, saveData, edit, update, these methods add bellow Line before return statements\*/

map.addAttribute(“uomTypes”,util.getUomTypes());

}

**3>In “UomRegister.jsp” and “UomDataEdit.jsp” :---**

=>remove <form:option value =“ “> tags, from 3 options “PACKING”, “NO-PACKING”,”NA” and replace with bellow line

<form:options items =”${uomType}”/>

**7>Implementing Rest Web Services application:--**

**1>Create One “RestController” class seprated if possible:--**

=>First completes all classes and interface as “UIContraller” and Add “RestController” class methods.

**package** com.uday.app.controller;

@RestController //org.springframework.web.bind.annotation.RestController;

@RequestMapping("rest/uom") //org.springframework.web.bind.annotation.RequestMapping;

**public** **class** UomRestController

{

@Autowired //org.springframework.beans.factory.annotation.Autowired;

**private** IUomService service;

@Autowired //org.springframework.beans.factory.annotation.Autowired;

**private** UomValidator validator;

}

**1>---------- Save UomData into Database:-------\*/**

@PostMapping("/save") //org.springframework.web.bind.annotation.PostMapping;

**public** ResponseEntity<?> save(@RequestBody Uom uom, Errors errors)

//org.springframework.web.bind.annotation.RequestBody;

//org.springframework.validation.Errors;

{

validator.validate(uom, errors); //com.uday.app.validator.UomValidator;

**if**(errors.hasErrors())

{

**return** ResponseEntity.*badRequest*().body(errors.getAllErrors());

//org.springframework.http.ResponseEntity;

}

**else**

{

Long uomId =service.save(uom);

**return** ResponseEntity.*ok*("Uom Created with Id :"+uomId);

}

}

**2>----Get All Data from Data Base Using “RestController” \*\*\*\*\*\*\*\*/**

@GetMapping("/all")

**public** ResponseEntity<Object> getAll()

{

Object response =**null**;

List<Uom> uomList=service.getAll();

**if**(uomList ==**null** || uomList.size() == 0)

{

response ="No Uom records Exist";

}

**else**

{

response = uomList;

}

**return** ResponseEntity.*ok*(response);

} ////http://localhost:2018/rest/ord/

**3>-----------Update Specific record By uomId -----\*\*\*\*\*\*\*\*\*/**

@PutMapping("/update") //org.springframework.web.bind.annotation.PutMapping;

**public** ResponseEntity<Object> update(@RequestBody Uom uom, Errors errors)

{

Object response =**null**;

validator.validate(uom, errors);

**if**(errors.hasErrors())

{

response = errors.getAllErrors();

**return** ResponseEntity.*badRequest*().body(response);

}

**else**

{

service.update(uom);

response ="Uom Updated with Id :" +uom.getUomId();

}

**return** ResponseEntity.*ok*(response);

}

//http://localhost:2018/rest/ord/update

**4>--------To Delete Specific records by using “uomId” ------\*\*\*\*\*\***

**i>Add bellow method in IUomService Interface.**

Public Boolean isExist(Long uomId);

**ii>Implement in “UomServiceImpl” class using repo.**

Public boolean isExist(Long ordId)

{

Return repo.exists(uomId);

}

**iii>Add bellow method in “UomRestController” class.**

@DeleteMapping("/delete/{uomId}") //org.springframework.web.bind.annotation.DeleteMapping;

**public** ResponseEntity<Object> delete(@PathVariable Long uomId)

//org.springframework.web.bind.annotation.PathVariable;

{

Object response =**null**;

**boolean** exist =service.isExist(uomId);

**if**(!exist)

{

response = "Uom '"+uomId+"' not exist";

}

**else**

{

service.delete(uomId);

response ="UomId '"+uomId+"' is Deleted";

}

**return** ResponseEntity.*ok*(response);

} //http://localhost:2018/rest/ord/delete/8

**Executions:---**

|  |  |  |
| --- | --- | --- |
| Post | http://localhost:2018/rest/uom/save | Send |
|  | Body | JSON (Application JSON) |

**8>Swagger with Spring Boot:---**

=>Swagger is a 3rd party API given by “Spring Fox”. It is used to generate HTML, UI for all our webServices that contains in Application.

=>It is implemented only once in Project (i.e not in every modules).

**1>Add bellow dependencies in pom.xml file:--**

<!-- Add two dependency 1>springFor-swagger2 2>springFox-swagger-ui for Swagger implementations -->

<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger2 -->

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger2</artifactId>

<version>2.7.0</version><scope>compiler</scope>

</dependency>

<!-- https://mvnrepository.com/artifact/io.springfox/springfox-swagger-ui -->

<dependency>

<groupId>io.springfox</groupId>

<artifactId>springfox-swagger-ui</artifactId>

<version>2.7.0</version>

<scope>compile</scope>

</dependency>

**2>Create a “SwaggerConfig” Class:--**

**package** com.uday.app.config;

@Configuration //org.springframework.context.annotation.Configuration;

@EnableSwagger2 //springfox.documentation.swagger2.annotations.EnableSwagger2;

**public** **class** SwaggerConfig

{

@Bean //org.springframework.context.annotation.Bean;

**public** Docket productAPI() //springfox.documentation.spring.web.plugins.Docket;

{

//springfox.documentation.spi.DocumentationType;

**return** **new** Docket (DocumentationType.***SWAGGER\_2***)

.select()

.apis(RequestHandlerSelectors.*basePackage*("com.uday.app"))

.paths(PathSelectors.*regex*("/rest/.\*"))

.build()

.apiInfo(metaData());

}

//Optional

**private** ApiInfo metaData() {

//springfox.documentation.service.ApiInfo;

//springfox.documentation.service.Contact;

ApiInfo apiInfo = **new** ApiInfo(

"Spring Boot REST API",

"Spring Boot REST API for Online Store",

"1.0","Terms of service",

**new** Contact("John Thompson", "https://springframework.guru/about/", "john@springfrmework.guru"),

"Apache License Version 2.0","https://www.apache.org/licenses/LICENSE-2.0");

**return** apiInfo; } }

**9>Pagination:--**

**1>Add below method in “IUomService” Interface:--**

**package** com.uday.app.service;

Public Page<Uom> findAll(Pageable p); //Pageable

**2>Implement above method in “UomServiceImpl” class:--**

**package** com.uday.app.service.impl;

@Override

**public** Page<Uom> findAll(Pageable p)

{

**return** repo.findAll(p);

}

**3>Add below code in “UomController” (Remove getAllRecords() method in controller and write this method:--**

**package** com.uday.app.controller;

@GetMapping("/all")

**public** String getAllRecords(@PageableDefault(size =4, sort="uomId",

direction=Direction.***DESC***)Pageable p, ModelMap map)

{

Page<Uom> page=service.findAll(p); //org.springframework.data.domain.Page;

map.addAttribute("page", page);

**return** "UomData";

}

**4>Add bellow code in “UomData.jsp”:--**

<html><body>

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

<div class=*"card-body"*>

<form:form action=*"all"* method=*"get"* modelAttribute=*"uom"* />

<c:forEach items=*"*${page.getContent()}*"* var=*"uom"*>

<tr>

<td><c:out value=*"*${uom.uomId}*"* /></td>

<td><c:out value=*"*${uom.uomType}*"* /></td>

<td><c:out value=*"*${uom.uomModel}*"* /></td>

<td><c:out value=*"*${uom.description}*"* /></td>

<td><c:out value=*"*${uom.createdOn}*"* /></td>

<td><c:out value=*"*${uom.modifiedOn}*"* /></td>

<td><a href=*"edit?uomId=*${uom.uomId}*"* class=*"btn btn-danger"*>Edit</a></td>

<td><a href=*"delete?uomId=*${uom.uomId}*"* class=*"btn btn-info"*>Delete</a></td>

</tr></c:forEach>

</table></div>

<!-- CARD BODY END -->

<!-- <Pagination Coding Starts> -->

<div class=*"card-footer text-white"*>

<ul class=*"pagination"*>

<c:if test=*"*${!page.isFirst()}*"*>

<li class=*"page-item"*><a class=*"page-link"* href=*"#"* onclick="setParam('page',0)">First</a></li>

</c:if>

<!-- End of First link -->

<c:if test=*"*${!page.hasPrevious()}*"*>

<li class=*"page-item"*><a class=*"page-link"* href=*"#"* onclick=*"setParam('page',*${page.getNumber()-1}*)"*>Previous</a></li>

</c:if>

<!-- End of Previous link -->

<c:if test=*"*${!**empty** page.getContent()}*"*>

<c:forEach begin=*"0"* end=*"*${page.getTotalPages()-1}*"* var=*"i"*>

<c:choose>

<c:when test=*"*${page.getNumber() **eq** i}*"*>

<li class=*"page-item active"*><a class=*"page-link"* href=*"#"* onclick=*"setParam('page',*${i}*)"*>${i+1}</a></li>

</c:when>

<c:otherwise>

<li class=*"page-item"*><a class=*"page-link"* href=*"#"* onclick=*"setParam('page',*${i}*)"*>${i+1}</a></li>

</c:otherwise>

</c:choose>

</c:forEach>

</c:if> <!-- End of Pagination Order -->

<c:if test=*"*${!page.hasNext()}*"*>

<li class=*"page-item"*><a class=*"page-link"* href=*"#"* onclick=*"setParam('page',*${page.getNumber()+1}*)"*>Next</a></li>

</c:if> <!-- End of Next link -->

<c:if test=*"*${!page.isLast()}*"*>

<li class=*"page-item"*><a class=*"page-link"* href=*"#"* onclick=*"setParam('page',*${page.getTotalPages()-1}*)"*>Last</a></li>

</c:if> <!-- End of Next link -->

</ul>

</div> <!-- Card Footer End /Pagination Coding ends-->

</div> <!-- CARD END -->

</div> <!-- COINTAINER END -->

</body> </html>

**10>Advanced Validations:--**

**1>Add bellow method in “UomRepository” (I):--**

**package** com.uday.app.repo;

/\*\*\*\*\*\*\*\*\*Advanced Validator Security checking \*\*\*\*\*\*\*\*\*\*\*\*\*/

@Query("select count(uomId) from com.uday.app.model.Uom where uomType=?1 and uomModel=?2")

**public** **long** countUomTypeAndModel(String uomType, String uomModel);

**2>Add bellow method in “IUomService” Intreface: ---**

**package** com.uday.app.service;

public boolean isUomTypeAndModelExist(String uomType, String uomModel);

**3>Add method in “UomServiceImpl” class: ---**

**package** com.uday.app.service.impl;

/\*\*\*\*\*\*\*\*\* Advanced Validator\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

@Override

**public** **boolean** isUomTypeAndModelExist(String uomType, String uomModel)

{

**long** count =repo.countUomTypeAndModel(uomType, uomModel);

**if**(count == 0)

**return** **false**;

**else**

**return** **true**;

}

**4>Create One “UomUtil” Class and add bellow methods:---**

**package** com.uday.app.util;

@Component //org.springframework.stereotype.Component;

**public** **class** UomUtil

{

**public** List<String> getUomTypes()

{

//1 method

**return** Arrays.*asList*("PACKING","NO-PACKING","NA");

//2 Method

/\*List<String> list = new ArrayList<String>();

{

list.add("PACKING");

list.add("No-PACKING");

list.add("NA");

return list;

}\*/

}

//Add all above List<String> getFieldName

**public** **void** addUiComponents(ModelMap map)

{

map.addAttribute("uomTypes", getUomTypes());

}

**5>Create one “Validator” class:---**

**package** com.uday.app.validator;

@Component //org.springframework.stereotype.Component;

**public** **class** UomValidator **implements** Validator //org.springframework.validation.Validator;

{

/\*2\*\*\*\*\*\*\*\*\*Advanced Validator\*\*\*\*\*\*\*\*\*\*\*\*/

@Autowired //HAS-A with uomUtil class

**private** UomUtil uomUtil;

@Autowired //HAS-A with uomService class

**private** IUomService uomService;

**private** **boolean** isEdit;

**public** **void** setEdit(**boolean** isEdit) {

**this**.isEdit = isEdit;

}

@Override

**public** **boolean** supports (Class<?> clazz)

{

**return** Uom.**class**.equals(clazz);

}

/\*1>\*\*\*\*\*\*\*\*\*Normal Validator\*\*\*\*\*\*\*\*\*\*\*/

/\*@Override

public void validate (Object target, Errors errors) //org.springframework.validation.Errors;

{

//ValidationUtils.rejectIfEmptyOrWhitespace(errors, "uomType", "uom.reg.type.error", "Enter defaultMessage");

//Direct supply message inside this

ValidationUtils.rejectIfEmptyOrWhitespace(errors, "uomType", "uom.reg.type.error");

ValidationUtils.rejectIfEmptyOrWhitespace(errors, "uomModel", "uom.reg.model.error");

ValidationUtils.rejectIfEmptyOrWhitespace(errors, "description", "uom.reg.desc.error");

}\*/

/\*2>\*\*\*\*\*\*Advanced Validator\*\*\*\*\*\*\*\*\*\*\*/

@Override

**public** **void** validate (Object target, Errors errors)

{

//Fields Validations

Uom uom =(Uom)target;

//List(Dropdown, CheckBox, RadioButton) fields Validation

**if**(!uomUtil.getUomTypes().contains(uom.getUomType()))

{

errors.rejectValue("uomType", "", " Please choose valid UomType");

}

//Advanced Fields Validations

**if**(!Pattern.*compile*("[A-Z \\t\\n\\x0B\\f\\r]{4,12}").matcher(uom.getUomModel()).matches())

{

errors.rejectValue("uomModel", "", "Enter 4-8 charecter in Capital letters Only, No Special Character Allowed");

}

**if**(!Pattern.*compile*("[a-zA-Z \\t\\n\\x0B\\f\\r]{10,250}").matcher(uom.getDescription()).matches())

{

errors.rejectValue("description", "", "Enter 10-250 charecter letters Only");

}

//Security Validation (Duplicate Data Validations) Server side validation

**if**(!isEdit && uomService.isUomTypeAndModelExist(uom.getUomType(), uom.getUomModel()))

{

errors.rejectValue("uomModel", "", "Uom '"+uom.getUomModel()+"' with '"+uom.getUomType()+"' exist");

} }

**6>Add/Modify “saveData” and “update” method in “UomController” class: ---**

**/\*2.1>\*\*\*\*\*\*\*\*\*\*Save date into DB after Validations\*\*\*\*\*\*\*\*\*\*\*\*\*/**

@PostMapping("/register")

**public** String saveData(@ModelAttribute Uom uom, Errors errors, ModelMap map)

{

/\*\*\*\*\*\*\*Server side validations\*\*\*\*/

validator.setEdit(**true**); //Its enable when we are not changing uomType and uomModel, i.e only for descriptions

validator.validate(uom, errors); //check for validation

**if**(!errors.hasErrors())

{

//if no errors,Save Data to DB using service Layer

Long uomId =service.save(uom);

//Clear form (model Object) after save

map.addAttribute("uom", **new** Uom());

map.addAttribute("message", "Uom Created with Id :"+uomId);

}

//getting Specific fields ListData(DropDown, CheckBox, RadioButton) from UI form

//map.addAttribute("uomTypes", util.getUomTypes());

//Getting(show) all ListData (DropDown, CheckBox, RadioButton) data from UI form

util.addUiComponents(map);

//UI PageName

**return** "UomRegister";

}

**/\*6>\*\*\*\*\*\*\*\*\*\*Update Specific record by uomId\*\*\*\*\*\*\*\*\*\*\*\*\*\*/**

@PostMapping("/update")

**public** String update(@ModelAttribute Uom uom,Errors errors, ModelMap map)

{

/\*\*\*\*\*\*\*Server side validations\*\*\*\*/

validator.setEdit(**true**); //Its enable when we are not changing uomType and uomModel, i.e only for descriptions

validator.validate(uom, errors); //check for validation

String page=**null**;

**if**(errors.hasErrors())

{

//map.addAttribute("uomTypes", util.getUomTypes());

util.addUiComponents(map);

page="UomDataEdit";

}

**else**

{

service.update(uom);

**return** "redirect:all";

}

**return** page;

}

**11>Multipart Controller:--**This controller is used to work with Bulk operations (Export/Import) huge data.

**1>Insert Bulk (Excel) Data into database:---**

**1>Add bellow details in pom.xml dependency.**

<!-- Add ""poi-ooxml" dependency for multipart controller implementations -->

<!-- https://mvnrepository.com/artifact/org.apache.poi/poi-ooxml -->

<dependency>

<groupId>org.apache.poi</groupId>

<artifactId>poi-ooxml</artifactId>

<version>3.9</version>

</dependency>

**2>Add bellow details in “application.properties” :--**

#Multipart Details

spring.http.multipart.max-file-size=10MB

spring.http.multipart.max-request-size=20MB

**3>Add one three (As much as variables apart from id, c\_date, m\_date) parameterized constructor in “Uom.java” file (Model class):-**

**package** com.uday.app.model;

//Create one 3 parameterized (As much as field available in model) constructor for multipart

**public** Uom(String uomType, String uomModel, String description)

{

**super**();

**this**.uomType = uomType;

**this**.uomModel = uomModel;

**this**.description = description;

}

**4>Add bellow method in “IUomService” interface to save List of Uoms :--**

**package** com.uday.app.service;

Public void save (List<Uom> uoms); //Multipart (Excel) Import

**5>Implements this method in “UomServiceImpl” class:--**

**package** com.uday.app.service.impl;

@Override

**public** **void** save(List<Uom> uoms)

{

repo.save(uoms);

}

**6>Define one method in “UomUtil” to convert multipartFile to List<Uom>:--**

**package** com.uday.app.util;

/\*\*\*\*\*\*\*\*\*Uom Multipart Data Import\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

//org.springframework.web.multipart.MultipartFile;

**public** List<Uom> processMultipart(MultipartFile uomFile)

{

List<Uom> uomList =**null**;

**if**(uomFile != **null**)

{

**try**

{

uomList =**new** ArrayList<Uom>();

InputStream is =uomFile.getInputStream(); //java.io.InputStream;

XSSFWorkbook book =**new** XSSFWorkbook(is); //org.apache.poi.xssf.usermodel.XSSFWorkbook;

XSSFSheet sheet =book.getSheet("uoms"); //org.apache.poi.xssf.usermodel.XSSFSheet;

Iterator<Row> rows =sheet.iterator(); //org.apache.poi.ss.usermodel.Row;

**while** (rows.hasNext())

{

Row row =rows.next();

**if**(row.getRowNum()==0)

**continue**;

Uom uom =**new** Uom(row.getCell(0).getStringCellValue(),

row.getCell(1).getStringCellValue(),

row.getCell(2).getStringCellValue());

uomList.add(uom);

} //While loop end

}

**catch**(IOException e)

{

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

}

} //if end

**return** uomList;

} //method end

}

**7>Create one “MultipartController” and add bellow method in “MultipartController” class: ---**

**package** com.uday.app.controller.multiparts;

@Controller

@RequestMapping("/uomMultiparts")

**public** **class** UomMultipartController

{

@Autowired

**private** UomUtil util; //com.uday.app.util.UomUtil;

@Autowired //org.springframework.beans.factory.annotation.Autowired;

**private** IUomService service; //com.uday.app.service.IUomService;

@GetMapping("/show")

**public** String showMultipartPage()

{

**return** "UomMultipart";

}

/\*2>\*\*\*\*\*Import Excel Data Into Database (On click Import Button) without alidation\*\*\*\*\*\*\*\*/

@PostMapping("/uomImport")

**public** String importUoms(@RequestParam MultipartFile uomFile, ModelMap map)

{

List<Uom> uomList =util.processMultipart(uomFile);

**if**(uomList.isEmpty())

{

map.addAttribute("message", "No Rows found");

}

**else**

{

service.save(uomList);

map.addAttribute("message", "Success");

}

**return** "UomMultipart";

}

**8> Create one “UomMultipart.jsp” file:--**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"* pageEncoding=*"ISO-8859-1"*%>

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

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

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html><head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<link rel=*"stylesheet"* href=*"../bootstrap/css/bootstrap-grid.min.css"*>

<link rel=*"stylesheet"* href=*"../bootstrap/css/bootstrap.min.css"*>

<script type=*"text/javascript"* src=*"../bootstrap/js/bootstrap.bundle.min.js"*></script>

<script type=*"text/javascript"* src=*"../bootstrap/js/bootstrap.min.js"*></script>

<script type=*"text/javascript"* src=*"../bootstrap/js/jquery-3.2.1.slim.min.js"*></script>

<script type=*"text/javascript"* src=*"../bootstrap/popper.min.js"*></script>

<title>Uom Multipart Import/Download Page</title>

</head>

<body>

<%-- <%@include file="Menu.jsp" %> --%>

<div class=*"container"*>

<div class=*"card"*>

<div class=*"Card-header bg-primary text-white"*>

<h2>WELCOME TO UOM Bulk Upload/Download Screen</h2>

</div> <!-- CARD HEAD END -->

<div class=*"card-body"*>

<form action=*"uomImport"* method=*"post"* enctype=*"multipart/form-data"*>

<div class=*"form-group"*>

<label for=*"uomFile"* class=*"control-label col-sm-2"*>Select File</label>

<input type=*"file"* name=*"uomFile"* class=*"custom-file-control col-sm-3"*/>

</div>

<input type=*"submit"* value=*"Import UOM's"* class=*"btn btn-success"*>

<a href=*"exportUoms"* class=*"btn btn-info"*>Export Data</a>

</form>

</div> <!-- CARD BODY END -->

<!-- Validation Errords Code Start-->

<c:if test=*"*${!**empty** mesage}*"*>

<div class=*"card-footer bg-success text-white"*>

<c:out value=*"*${message}*"*></c:out>

</div> <!-- CARD FOOTER END -->

</c:if>

<c:if test =*"*${!**empty** errorMap}*"*>

<table class=*"table table-hover"*>

<tr class=*"thread-light"*>

<th>#RowNo</th>

<th>Errors</th>

</tr>

<c:forEach items=*"*${errorMap}*"* var=*"e"*>

<tr> <td><c:out value=*"*${e.key}*"*></c:out></td>

<td><c:out value=*"*${e.value}*"*></c:out></td>

</tr> </c:forEach>

</table>

</c:if>

<!-- Validation Eroors code Ends -->

</div> <!-- CARD END -->

</div> <!-- CONTAINER HEAD END -->

</body></html>

**9>Create one “UomMultipartValidator” class:--** This validator is designed to check errors in Uploaded Excel sheet.

i.e works on List<Uom>. It returns Error (if exist) in “Map<String, List<String>>”.

**package** com.uday.app.controller.multipart.validator;

@Component

**public** **class** UomMultipartValidator

{

@Autowired

**private** UomUtil util;

@Autowired

**private** IUomService service;

**public** Map<String, List<String>> validateUoms(List<Uom> uoms)

{

Map<String, List<String>> errorsMap=**new** LinkedHashMap<String, List<String>>();

**int** i =1;

**for**(Uom uom:uoms)

{

List<String> errorsList =**new** ArrayList<String>();

//Check one by one Object

**if**(StringUtils.*isEmpty*(uom.getUomType()))

{

errorsList.add("UomType can't be empty");

}

**else** **if**(!util.getUomTypes().contains(uom.getUomType()));

{

errorsList.add("uomType must be one of :"+util.getUomTypes().toString());

}

**if**(StringUtils.*isEmpty*(uom.getUomModel()))

{

errorsList.add("UomModel Can't be Empty");

}

**else** **if**(!Pattern.*compile*("[A-Z]{4,8}").matcher(uom.getUomModel()).matches())

{

errorsList.add("UomModel must be 4-8 upper case letters");

}

**if**(StringUtils.*isEmpty*(uom.getDescription()))

{

errorsList.add("Description can't be empty");

}

**else** **if**(!Pattern.*compile*("[a-zA-Z]{10,250}").matcher(uom.getDescription()).matches());

{

errorsList.add("Description must be 10-250");

}

**if**(service.isUomTypeAndModelExist(uom.getUomType(), uom.getUomModel()))

{

errorsList.add("Uom '"+uom.getUomModel()+"'alteast exist with type'"+uom.getUomType()+"'");

}

**if**(!errorsList.isEmpty())

errorsMap.put("Errors at Row#"+i, errorsList);

i++;

}

**return** errorsMap;

}

**10>Create one “UomMultipartController” class:--**Replace code of “importUoms methods” with given one.

=>Make Has-A relation between UomMultipartController-------------<>UomMultipartValidator.

**package** com.uday.app.controller.multiparts;

@Controller

@RequestMapping("/uomMultiparts")

**public** **class** UomMultipartController

{

@Autowired

**private** UomUtil util; //com.uday.app.util.UomUtil;

@Autowired //org.springframework.beans.factory.annotation.Autowired;

**private** IUomService service; //com.uday.app.service.IUomService;

@Autowired

**private** UomMultipartValidator validator; //Has-A relation with Validator Class

@GetMapping("/show")

**public** String showMultipartPage()

{

**return** "UomMultipart";

}

@PostMapping("uomImport")

**public** String importsUoms(@RequestParam MultipartFile uomFile, ModelMap map)

{

//Check uploaded one is Valid file

**if**(uomFile == **null** || !uomFile.getOriginalFilename().contains(".xlsx"))

{

map.addAttribute("message", "Choose Invalid File");

}

**else**

{

List<Uom> uomList = util.processMultipart(uomFile);

**if**(uomList.isEmpty())

{

map.addAttribute("message", "No rows found in Sheet");

}

**else**

{

Map<String, List<String>> errorMap = validator.validateUoms(uomList);

**if**(errorMap.isEmpty())

{

service.save(uomList);

map.addAttribute("message", "File Uoms Data uploaded Successfully");

}

**else**

{

map.addAttribute("errorMap", errorMap);

} } }

**return** "UomMultipart";

}

**11>Add below code in “UomMultipart.jsp”:-**-

<!-- Validation Errors Code Start-->

<c:if test=*"*${!**empty** mesage}*"*>

<div class=*"card-footer bg-success text-white"*>

<c:out value=*"*${message}*"*></c:out>

</div> <!-- CARD FOOTER END -->

</c:if>

<c:if test =*"*${!**empty** errorMap}*"*>

<table border =*"1"* class=*"table table-hover"*>

<tr class=*"thread-light"*>

<tr>

<th>#RowNo</th>

<th>Errors</th>

</tr>

<c:forEach items=*"*${errorMap}*"* var=*"e"*>

<tr>

<td><c:out value=*"*${e.key}*"*></c:out></td>

<td><c:out value=*"*${e.value}*"*></c:out></td>

</tr>

</c:forEach>

</table>

</c:if>

<!-- Validation Errors code Ends -->

**2>Export/Download data from DataBase in Excel Format:--**

**1>Add one method in “UomMultipartController.java”:---**

**package** com.uday.app.controller.multiparts;

/\*2>\*\*\*\*\*\*\*Download(Export) data from Database in Excel Sheet\*\*\*\*\*\*\*\*\*/

@RequestMapping("exportUoms")

**public** ModelAndView exportData()

{

List<Uom> uoms =service.getAll();

**return** **new** ModelAndView (**new** UomXlsxView(),"uomList",uoms);

}

**2>Create one new Class as “UomXlsxView”:--- It extends AbstractXlsxView Class**

**package** com.uday.app.view;

**public** **class** UomXlsxView **extends** AbstractXlsxView // org.springframework.web.servlet.view.document.AbstractXlsxView;

{

@Override

**protected** **void** buildExcelDocument(Map<String, Object> model, Workbook workbook, HttpServletRequest request,

HttpServletResponse response) **throws** Exception

/\*org.apache.poi.ss.usermodel.Workbook;

//javax.servlet.http.HttpServletRequest; javax.servlet.http.HttpServletResponse;\*/

{

//Set Book name (Excel FileName)

response.setHeader("Content-Disposition", "attachment;filename=uomdata.xlsx");

//1>Read Data

@SuppressWarnings("unchecked")// optional

List<Uom> uoms = (List<Uom>)model.get("uomList");

//3>Create Sheet

Sheet sheet = workbook.createSheet("UOMs"); //org.apache.poi.ss.usermodel.Sheet;

//3>set Head

setHead(sheet);

//Set Body

setBody(sheet,uoms);

}

**private** **void** setHead (Sheet sheet)

{

Row row =sheet.createRow(0); //org.apache.poi.ss.usermodel.Row;

row.createCell(0).setCellValue("ID");

row.createCell(1).setCellValue("TYPE");

row.createCell(2).setCellValue("CODE");

row.createCell(3).setCellValue("DESC");

row.createCell(4).setCellValue("CREATED");

row.createCell(5).setCellValue("MODIFIED");

}

**private** **void** setBody(Sheet sheet, List<Uom> uoms)

{

**int** rowNum =1;

**for**(Uom uom:uoms)

{

Row row =sheet.createRow(rowNum++);

row.createCell(0).setCellValue(uom.getUomId());

row.createCell(1).setCellValue(uom.getUomType());

row.createCell(2).setCellValue(uom.getUomModel());

row.createCell(3).setCellValue(uom.getDescription());

row.createCell(4).setCellValue(uom.getCreatedOn().toString());

row.createCell(5).setCellValue(uom.getModifiedOn().toString());

}

}

}

**12>Specification :---**

**1>Repository interface must implements “JpaSpecificationExecutor<T>”:--**

**package** com.uday.app.repo;

**public** **interface** UomRepository **extends** JpaRepository<Uom, Long>,JpaSpecificationExecutor<Uom>{

}

**2>In “IUomService” modify getAll()/findAll() method with specification Parameters:--**

**package** com.uday.app.service;

**public** Page<Uom> findAll(Specification<Uom> spec, Pageable page); //Specification

**3>In UomServiceImpl modify method as:--**

**package** com.uday.app.service.impl;

/\*\*\*\*\*\*\*\*\*Specification\*\*\*\*\*\*\*\*\*\*\*/

@Override

**public** Page<Uom> getAll(Specification<Uom> spec, Pageable page)

{

**return** repo.findAll(spec, page);

}

**4>Create one “UomSpecification” class that implements Specification (I) interface:--**

**package** com.uday.app.spec;

//org.springframework.data.jpa.domain.Specification;

**public** **class** UomSpecification **implements** Specification<Uom>

{

**private** Uom filter;

**public** UomSpecification(Uom uom)

{

**this**.filter =uom;

}

@Override

**public** Predicate toPredicate(Root<Uom> root, CriteriaQuery<?> query, CriteriaBuilder cb)

{

Predicate p =cb.conjunction();//javax.persistence.criteria.Predicate;

//1. Methods

**if**(!StringUtils.*isEmpty*(filter.getUomType()))

{

p.getExpressions().add(cb.equal(root.get("uomType"),filter.getUomType()));

}

**if**(!StringUtils.*isEmpty*(filter.getUomModel()))

{

p.getExpressions().add(cb.like(root.get("uomModel").as(String.**class**),"%"+filter.getUomModel()+"%"));

}

**if**(!StringUtils.*isEmpty*(filter.getDescription()))

{

p.getExpressions().add(cb.like(root.get("description").as(String.**class**),"%"+filter.getDescription()+"%"));

}

**return** p;

}

}

**5>Add bellow methods in “UomController” class:---**

=>Modify pagination method as bellow

/\*\*\*\*\*\*\*\*\*\*Specification\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

@GetMapping("/all")

**public** String getUoms(@PageableDefault(size=5, sort="uomId", direction=Direction.***ASC***) Pageable p,

@ModelAttribute Uom uom, ModelMap map)

{

UomSpecification spec = **new** UomSpecification(uom);

Page<Uom> page =service.getAll(spec, p);

map.addAttribute("page", page);

map.addAttribute("uom", uom);

//Show Specific(uomType) listData(DropDown) on UI form

//map.addAttribute("uomTypes", util.getUomTypes()); //Show only uomType data

util.addUiComponents(map); //Show all ListBox(DropDown, Radiobutton, CheckBox) data

**return** "UomData";

}

**6>Define one Search (spring form) screen in “UomData.jsp” and also add<Script> given bellow:--**

=>Modify pagination code lines as bellow.

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"* pageEncoding=*"ISO-8859-1"*%>

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

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

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

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<link rel=*"stylesheet"* href=*"../bootstrap/css/bootstrap-grid.min.css"*>

<link rel=*"stylesheet"* href=*"../bootstrap/css/bootstrap.min.css"*>

<script type=*"text/javascript"*

src=*"../bootstrap/js/bootstrap.bundle.min.js"*></script>

<script type=*"text/javascript"* src=*"../bootstrap/js/bootstrap.min.js"*></script>

<script type=*"text/javascript"*

src=*"../bootstrap/js/jquery-3.2.1.slim.min.js"*></script>

<script type=*"text/javascript"* src=*"../bootstrap/js/popper.min.js"*></script>

<script type=*"text/javascript"* src=*"../bootstrap/js/SearchPageParam.js"*></script>

<title>Uom Data Page</title>

</head>

<body>

<%-- <%@include file="Menu.jsp"%> --%>

<div class=*"container"*>

<div class=*"card"*>

<div class=*"card-header bg-info text-white"*>

<h1>Welcome to UomData</h1>

</div> <!-- CARD HEAD END -->

<!-- Specification form Code starts -->

<div class=*"card-body"*>

<form:form cssClass=*"form-inline"* action=*"all"* method=*"get"* modelAttribute=*"uom"*>

<div class=*"form-group"*>

<label for=*"uomType"* class=*"control-label col-sm-3"*>Uom Type</label>

<form:select path=*"uomType"* cssClass=*"form-control col-sm-7"*>

<form:option value=*""*>--Select--</form:option>

<form:options items=*"*${uomTypes}*"* />

</form:select><br>

</div>

<div class=*"form-group"*>

<label for=*"uomModel"* class=*"control-label"*>Uom Model</label>

<form:input path=*"uomModel"* cssClass=*"form-control col-sm-7"*/><br>

</div>

<div class=*"form-group"*>

<label for=*"description"* class=*"control-label col-sm-4"*>Description</label>

<form:textarea path=*"description"* cssClass=*"form-control col-sm-7"*/><br>

</div>

<input type=*"submit"* value=*"Search UOM's"* class=*"btn btn-success"*/>

</form:form>

<br> <!-- Specifiction Form code ends -->

<table class=*"table table-hover"*>

<tr class=*"thread-light"*>

<th>ID</th>

<th>TYPE</th>

<th>MODEL</th>

<th>DESC</th>

<th>CREATED</th>

<th>MODIFIED</th>

<th colspan=*"2"*>OPERATIONS</th>

</tr>

<%-- <c:forEach items="${uomList}" var="ord"> --%>

<c:forEach items=*"*${page.getContent()}*"* var=*"uom"*>

<tr>

<td><c:out value=*"*${uom.uomId}*"* /></td>

<td><c:out value=*"*${uom.uomType}*"* /></td>

<td><c:out value=*"*${uom.uomModel}*"* /></td>

<td><c:out value=*"*${uom.description}*"* /></td>

<td><c:out value=*"*${uom.createdOn}*"* /></td>

<td><c:out value=*"*${uom.modifiedOn}*"* /></td>

<td><a href=*"edit?uomId=*${uom.uomId}*"* class=*"btn btn-danger"*>Edit</a></td>

<td><a href=*"delete?uomId=*${uom.uomId}*"* class=*"btn btn-info"*>Delete</a></td>

</tr>

</c:forEach>

</table>

</div> <!-- CARD BODY END -->

<!-- <Pagination Coding Starts> -->

<div class=*"card-footer text-white"*>

<ul class=*"pagination"*>

<c:if test=*"*${!page.isFirst()}*"*>

<li class=*"page-item"*><a class=*"page-link"* href=*"#"* onclick="setParam('page',0)">First</a></li>

</c:if> <!-- End of First link -->

<c:if test=*"*${!page.hasPrevious()}*"*>

<li class=*"page-item"*><a class=*"page-link"* href=*"#"* onclick=*"setParam('page',*${page.getNumber()-1}*)"*>Previous</a></li>

</c:if> <!-- End of Previous link -->

<c:if test=*"*${!**empty** page.getContent()}*"*>

<c:forEach begin=*"0"* end=*"*${page.getTotalPages()-1}*"* var=*"i"*>

<c:choose>

<c:when test=*"*${page.getNumber() **eq** i}*"*>

<li class=*"page-item active"*><a class=*"page-link"* href=*"#"* onclick=*"setParam('page',*${i}*)"*>${i+1}</a></li>

</c:when>

<c:otherwise>

<li class=*"page-item"*><a class=*"page-link"* href=*"#"* onclick=*"setParam('page',*${i}*)"*>${i+1}</a></li>

</c:otherwise>

</c:choose>

</c:forEach>

</c:if>

<c:if test=*"*${!page.hasNext()}*"*>

<li class=*"page-item"*><a class=*"page-link"* href=*"#"*

onclick=*"setParam('page',*${page.getNumber()+1}*)"*>Next</a></li>

</c:if> <!-- End of Next link -->

<c:if test=*"*${!page.isLast()}*"*>

<li class=*"page-item"*><a class=*"page-link"* href=*"#"*

onclick=*"setParam('page',*${page.getTotalPages()-1}*)"*>Last</a></li>

</c:if>

<!-- End of Last link -->

</ul>

</div> <!-- Card Footer End /Pagination Coding ends-->

<!-- End of Pagination Code -->

</div> <!-- CARD END -->

</div> <!-- COINTAINER END -->

</body>

</html>

Spring API for Email :--- Sun has provided basic Mail API also called as Java Mail API. This Java Mail API is built using complex Email Structure.

=>This is simplified using Spring Email API by using POJO-POJI Design Pattern.

=>It also provides one helper class “MimeMessageHelper” which is used to write code in less line (easy coding).

=>It supports MIME messages using Helper class configuration MIME.stand for Multipurpose Internet Mail Extension i.e We can send data using simple or any attachment to multiple destinations using To, CC(Carbon copy), BCC(Black Carbon copy).

1>Add “mail” and “spring-integration-mail” jars in lib folder.

<dependency>

<groupId>javax.mail</groupId>

<artifactId>mail</artifactId>

<version>1.4.3</version>

</dependency>

<dependency>

<groupId>org.springframework.integration</groupId>

<artifactId>spring-integration-mail</artifactId>

<version>4.2.3.RELEASE</version>

</dependency>

2>Add bellow method in “IUomService” Intreface: ---

3>Add method in “UomServiceImpl” class: ---

4>Add method in Controller class: ---

**Location Data Export to Excel/Pdf :--**

=>Spring f/w has provided AbstractView class to export data to Excel or PDF format.

=>Spring f/w used Apache POI and provided “AbstractExcelView” (Abstract class).

=>Use above pre-defined PDF classes and define child classes by extending them.

=>Use AbstractExcelView for excel file export and AbstractPdfView for PDF file.

=>Use XmlViewResolver to identify ExcelView class or PdfView class based on input given

by controller method with order =0.

=>If XmlViewResolver is unable to find any view class then request is given to InternalResourceViewResolver.

=>All Excel and Pdf view class must be configured in spring-view.xml file which is defined by XmlViewResolver.

=>Share data to view class from controller using ModelMap.

=>To call these defined HyperLinks at UI like Location Excel/Location Pdf.

**Data Excel Export format :--**

=>One excel file is called as WorkBook (extension is **.xls**, it should not **.xlsx.)**

=>Every Book contains sheets identified by unique Name in Book.

=>Sheet can have rows starts from zero.

=>Every Row will be having cells (Boxes) starts from zero.

=>To create these in Java Application use API given by spring with Apache POI named as HSSF(Horrible Spread Sheet Format).

**API Classes are:--**

1>HSSFWorkbook = Excel File

2>HSSFSheet =Excel Sheet

3>HSSRow =Row of Sheet

4>HSSCell =Cell of Row

NOTE:-- Workbook will be provided (created ) by Spring f/w, user should create sheet with Rows and cells only.

**A>Excel Export:--**

**1>Create one View class with name “LocationExcelView” under src and define code for Excel export like:--**

com.app.view;

**public** **class** LocationExcelView **extends** AbstractExcelView //org.springframework.web.servlet.view.document.AbstractExcelView;

{

@Override //org.apache.poi.hssf.usermodel.HSSFWorkbook;

**public** **void** buildExcelDocument (Map<String, Object>map, HSSFWorkbook book,

HttpServletRequest req, HttpServletResponse res) **throws** Exception

{

//File Name

res.addHeader("Content-Disposition", "attachment;filename=LOCATION.xls");

//1. Read data from ModelMap

List<Location> locs =(List<Location>)map.get("locs");

//2. Create Sheet

HSSFSheet sheet=book.createSheet("loc");

//3Construct Header

setHead(sheet);

//4Construct Body

setBody(sheet, locs);

}

**private** **void** setHead(HSSFSheet sheet) //org.apache.poi.hssf.usermodel.HSSFSheet;

{

HSSFRow row=sheet.createRow(0); //org.apache.poi.hssf.usermodel.HSSFRow;

row.createCell(0).setCellValue("ID");

row.createCell(1).setCellValue("NAME");

row.createCell(2).setCellValue("TYPE");

}

**private** **void** setBody(HSSFSheet sheet, List<Location> locs)

{

**int** rowNum=1;

**for**(Location loc : locs)

{

HSSFRow row =sheet.createRow(rowNum++);

row.createCell(0).setCellValue(loc.getLocId());

row.createCell(1).setCellValue(loc.getLocName());

row.createCell(2).setCellValue(loc.getLocType());

}

}

}

**2>Create “spring-views.xml” file under “WEB-INF” copy <beans> tag code from spring 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"*

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

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

<bean name=*"LocExcel"* class=*"com.app.view.LocationExcelView"*/>

<bean name=*"LocPdf"* class=*"com.app.view.LocationPdfView"*/>

</beans>

**3>add bellow code in “sample-servlet.xml” (spring configuration file at bottom):--**

<!-- XML ViewResolver -->

<bean class=*"org.springframework.web.servlet.view.XmlViewResolver"*>

<property name=*"location"*>

<value>/WEB-INF/spring-views.xml</value>

</property>

<property name=*"order"* value=*"0"* />

</bean>

**4>Add bellow method in “LocationController” class:--**

//7.Export data to Excel

@RequestMapping("/locExcelExport")

//to send the data form modelMap to UI

**public** String exporttoExcel(ModelMap map)

{

// get all the data form DB and save the data on one key

// call the service call and the call the getAllLocations method

map.addAttribute("locs", service.getAllLocations());

// pass the data to the spring-view.xml page if match and get the class name and

// search the class which have logic to export the Excel

**return** "locExcel";

}

**5>Add bellow Hyperlink in “LocationData.jsp”:--**

<a href=*"locExcelView"*>Export Excel Data</a>

**b>PDF EXPORT:---**

**Create elements :--** Paragraph Image, table, Header/Footer, Text, file etc.. and add to Document.

NOTE:-- If not added to document these will not be displayed at PDF file.

NOTE:-- Refer <http://developers.itextpdf.com/examples/itext-action-second-edition/chapter-1>.

**1>Create one class with name “LocationPdfView” under src and define code for Excel export like.**

**package** com.app.view;

//org.springframework.web.servlet.view.document.AbstractPdfView;

**public** **class** LocationPdfView **extends** AbstractPdfView

{

@Override

//com.lowagie.text.Document; //com.lowagie.text.pdf.PdfWriter;

//javax.servlet.http.HttpServletRequest; //javax.servlet.http.HttpServletResponse;

**protected** **void** buildPdfDocument(Map<String, Object> map, Document document, PdfWriter pdfw,

HttpServletRequest req, HttpServletResponse res) **throws** Exception

{

//logic

document.add(**new** Paragraph("Welcome to PDF")); //com.lowagie.text.Paragraph;

//Read Data

List<Location> locList =(List<Location>)map.get("locListObj");

PdfPTable table =**new** PdfPTable(3); //com.lowagie.text.pdf.PdfPTable;

table.addCell("Location Id");

table.addCell("Location Name");

table.addCell("Location Type");

**for**(Location loc:locList)

{

table.addCell(""+loc.getLocId());

table.addCell(loc.getLocName());

table.addCell(loc.getLocType());

}

document.add(table);

}

}

**2>Configure above class in “spring-view.xml” as a <bean>.**

Ex:-- <bean name=*"LocPdf"* class=*"com.app.view.LocationPdfView"*/>

**3>Define one controller method for Location PDF Export call in Location controller class.**

**package** com.app.controller;

/\*\*\*\*\*\*\* Export Data into PDF\*\*\*\*\*\*\*/

@RequestMapping("/locPdfView")

**public** String generatePdf(ModelMap map)

{

map.addAttribute("locListObj", service.getAllLocations());

**return** "LocPdf";

}

**4>Add Hyperlink at “LocationData.jsp”.**

<a href=*"locPdfView"*>Export PDF</a>

**JFreeCharts using Java and Spring F/W:--**

**=>Location data** stored in DB with table name “LOCTAB” which are two types Urban and Rural. This data can be shown in graphical format using JFreeChart using PIE and BAR charts.

=>For this prepared data as

1>no of Rural = <count>

2>no of Urban = <count>

SQL Query looks like : select loctype, count(lid) from loctab group by ltype;

=>Application will not accept SQL, so it must be converted to HQL.

=>To convert SQL to HQL in query replace tablename with ModelCassName and collumnName with variableName then HQL query looks like this.

=>select locType, count(locId) from com.app.model.Location group by locType;

=>Execute above HQL query using ht.find(hql); that returns List<Object[]>.

=>JFree find method is used to execute HQLs to get partial data from DB. (Use get or loadAll for full data).

**Chart implementation code:---**

**1>Add one method in “ILocationDao”:--**

/\*6>\*\*\*get number of Count based on specific column (JfreeChart )\*\*\*\*\*/

**public** List<Object[]> getLocTypeWiseCount();

**2>Implement method in “LocationDaoImpl”:--**

/\*6>\*\*\*get number of Count based on specific column\*\*\*\*\*/

@Override

**public** List<Object[]> getLocTypeWiseCount() {

String hql="select locType,count(locType) "

+" from "+Location.**class**.getName()

+" group by locType ";

List<Object[]> lst=ht.find(hql);

**return** lst;

}

**3>Copy above ”ILocationDao” PIE/BAR chart method and past in “ILocationService” (Or add below method in ILocationService):--**

**public** List<Object[]> getLocTypeWiseCount(); //get(count) specific column data

**4>Implement method in “LocationServiceImpl”:--**

/\*6>\*\*\*get number of Count based on specific column\*\*\*\*\*/

@Override

**public** List<Object[]> getLocTypeWiseCount() {

**return** dao.getLocTypeWiseCount();

}

**5>Define one “LocationUtil.java” class:--**

com.app.util;

@Component //org.springframework.stereotype.Component;

**public** **class** LocationUtil

{

@Autowired //org.springframework.beans.factory.annotation.Autowired;

**private** ILocationService service;

**public** List<Location> getAllLocations() {

**return** service.getAllLocations();

}

**public** **void** generatePie(String path,List<Object[]> data){

//1.convert to dataset

DefaultPieDataset dataset=**new** DefaultPieDataset(); //org.jfree.data.general.DefaultPieDataset;

**for**(Object[] ob:data){

dataset.setValue(ob[0].toString(), **new** Double(ob[1].toString()));

}

//2.convert dataset to JFreeChart obj

//org.jfree.chart.JFreeChart;, org.jfree.chart.ChartFactory;

JFreeChart chart=ChartFactory.*createPieChart3D*("Location Report", dataset, **true**,**true**, **false**);

//3.convert to image

**try** { //java.io.File;

ChartUtilities.*saveChartAsJPEG*(**new** File(path+"/reportALoc.jpg"), chart, 400,300); //org.jfree.chart.ChartUtilities;

} **catch** (Exception e) {

e.printStackTrace();

}

}

**public** **void** generateBar(String path,List<Object[]> data){

//1.create dataset

DefaultCategoryDataset dataset=**new** DefaultCategoryDataset(); //org.jfree.data.category.DefaultCategoryDataset;

**for**(Object[] ob:data){

dataset.setValue(**new** Double(ob[1].toString()), ob[0].toString(),"");

}

//convert to JFreeChart

JFreeChart chart=ChartFactory.*createBarChart3D*("Location Report", "Location Type", "Location Count", dataset);

//convert to images

**try** {

ChartUtilities.*saveChartAsJPEG*(**new** File(path+"/reportBLoc.jpg"), chart, 400, 400);

} **catch** (Exception e) {

e.printStackTrace();

}

}

}

**6>Define “LocationContoller” HAS-A with “ServiceContext” and “LocationUtil”, also add method:--**

com.app.controller;

// 9 BAR/PIE Chart For Location

@RequestMapping("/locBarReports")

**public** String showBarCharts(){

String path=context.getRealPath("/");

List<Object[]> data=service.getLocTypeWiseCount();

util.generateBar(path, data); //Display Bar report

util.generatePie(path, data); //Display Pie chart report

**return** "LocationReport";

}

**7>Create “LocationReport.jsp” in views.**

<img src="../reportALoc.jpg"/>

<img src="../reportBLoc.jpg"/>

**Document Upload and DownLoad:---**

**1>Configure “MultipartResolver” in “sample-servlet.xml”:--**

<!-- Multipart Enable -->

<bean class=*"org.springframework.web.multipart.commons.CommonsMultipartResolver"* name=*"multipartResolver"*>

<property name=*"maxUploadSize"* value=*"20971520"* />

<property name=*"maxInMemorySize"* value=*"10485760"* />

</bean>

**2>Create a “Document” Model class:--**

com.app.model;

@Entity

@Table(name="appdoctab")

public class Document

{

@Id

@Column(name ="fId")

private int fileId;

@Column(name="fName")

private String fileName;

@Lob

@Column(name="fData")

private byte[] fileData;

}

**3>Add one method in “IDocumentDao”:--**

com.app.dao;

public void saveDocument(Document doc);

public List<Object[]> getFileIdAndNames();

public Document getDocById(int docId);

**4>Implement method in “DocumentDaoImpl”:--**

com.app.dao.impl;

@Repository

public class DocumentDaoImpl implements IDocumentDao

{

@Autowired

private HibernateTemplate ht;

//Save Operation

@Override

public void saveDocument(Document doc) {

ht.save(doc);

}

@Override

public List<Object[]> getFileIdAndNames()

{

String hql ="select fileId, fileName from "+Document.class.getName();

List<Object[]> list =ht.find(hql) ;

return list;

}

//Download data from database

@Override

public Document getDocById(int docId)

{

return ht.get(Document.class, docId);

} }

**5>Copy above ”IDocumentDao” method and past in “IDocumentService” (Or add below method in IDocumentService):--**

com.app.service;

public void saveDocument(Document doc);

public List<Object[]> getFileIdAndNames();

public Document getDocById(int docId);

**6>Implement method in “DocumentServiceImpl”:--**

com.app.service.impl;

@Service

public class DocumentServiceImpl implements IDocumentService

{

@Autowired

private IDocumentDao dao;

//Show Document upload page

@Override

public void saveDocument(Document doc)

{

dao.saveDocument(doc);

}

@Override

public List<Object[]> getFileIdAndNames()

{

return dao.getFileIdAndNames();

}

//Download Documents

@Override

public Document getDocById(int docId)

{

return dao.getDocById(docId);

}

}

**7>Define one “DocumentContoller” HAS-A with “IDocumentService” also add method:--**

com.app.controller;

@Controller

public class DocumentController

{

@Autowired

private IDocumentService service;

/\*1>\*\*\*\*\*\*\*\*Show Upload page with data\*\*\*\*\*\*\*\*\*\*\*/

@RequestMapping("showDocs")

public String showPage(ModelMap map)

{

List<Object[]> list=service.getFileIdAndNames();

map.addAttribute("docsData", list);

return "MyDocuments";

}

/\*2>To Upload Data into database \*\*\*\*\*\*\*\*\*\*\*\*\*/

@RequestMapping(value="insertDoc", method=RequestMethod.*POST*)

//org.springframework.web.multipart.commons.CommonsMultipartFile;

public String saveDoc(@RequestParam("fId") int fileId, @RequestParam("fileObj")CommonsMultipartFile cmfile )

{

if(cmfile !=null)

{

Document doc = new Document();

//convert UI data to model obj

doc.setFileId(fileId);

doc.setFileName(cmfile.getOriginalFilename());

doc.setFileData(cmfile.getBytes());

//Save Model Object using servie

service.saveDocument(doc);

}

return "redirect:showDocs";

}

//To Download File(Data)

@RequestMapping("/downloadFile")

public void downloadDoc(@RequestParam("fileId") int docId, HttpServletResponse res)

{

//get Document obj by id

Document doc =service.getDocById(docId);

//add "Download header"

//Content-Disposition

res.addHeader("Content-Disposition", "attachment;filename="+doc.getFileName());

//copy data to output stream from file

try

{

FileCopyUtils.*copy*(doc.getFileData(), res.getOutputStream()); //org.springframework.util.FileCopyUtils;

}

catch (IOException e)

{

e.printStackTrace();

}

}

}

**8>Create “MyDocuments.jsp” in views:--**

<%@ page language=*"java"* contentType=*"text/html; charset=ISO-8859-1"*

pageEncoding=*"ISO-8859-1"*%>

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

<!DOCTYPE html PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/TR/html4/loose.dtd">

<html>

<head>

<meta http-equiv=*"Content-Type"* content=*"text/html; charset=ISO-8859-1"*>

<title>Insert title here</title>

</head>

<body>

<%-- <%@include file="Master.jsp" %> --%>

<h1>Welcome to Documents Page</h1>

<form action=*"insertDoc"* method=*"post"* enctype=*"multipart/form-data"*>

<pre>

id : <input type=*"text"* name=*"fId"*/><br>

file: <input type=*"file"* name=*"fileObj"*/><br>

<input type=*"submit"* value=*"Upload"*/>

</pre>

</form>

<table border=*"1"*>

<tr>

<th>Id</th>

<th>File</th>

<th>Link</th>

</tr>

<c:forEach items=*"*${docsData}*"* var=*"ob"*>

<tr>

<td><c:out value=*"*${ob[0]}*"*/></td>

<td><c:out value=*"*${ob[1]}*"*/></td>

<td><a href=*'downloadFile?fileId=*<c:out value="${ob[0]}"/>*'*><img src=*"../images/download.jpg"*, height =*"25"*, width=*"25"*/></a></td>

</tr>

</c:forEach>

</table>

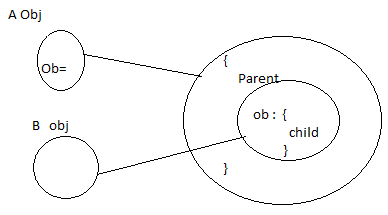
</body>

</html>

**Integrations :--**

HAS-A Relations :-- Using child class or interface as DataType in parent class and creating variable is called as HAS-A relation.

=>In case of JSON conversion, for HAS-A relation objects inner JSON will be created for child type in parent JSON. It is looks like



**Modules Integration :--**

1>Create HAS-A relation between two classes (Customer……………..<> Location).

2>Apply multiplicity annotation (1..1, 1…\*, \*…1, \*…\*) also generate set/get—toString

3>Make HAS-A relation (ParentUtil……<>Child IService)

Ex:-- CustomerUtil…………………<>ILocationService

4>Call getAll() method using child service and add data to ModelMap in same Util.

5>AT JSP use <c:forEach and <option> to Display data as Dropdown.

**1>Using Child class as DataType create one variable in Parent class:--**

com.app.model;

@ManyToOne //javax.persistence.ManyToOne;

@JoinColumn(name="locIdFk") //javax.persistence.JoinColumn;

**private** Location loc;

//Generate setters and Getters and toString methods

**2>Add bellow methods in “CustomerUtil” :--**

@Autowired

**private** ILocationService service;

**public** List<String> custTypeSelect()

{

List<String> listSelect=Arrays.*asList*("SELLER", "CUNSUMER");

**return** listSelect;

}

**public** **void** addToUi(ModelMap map)

{

map.addAttribute("custType", custTypeSelect());

map.addAttribute("locsUi", service.getAllLocations());

}

3>Make has a relation with CustomerController……………<>CustomerUtil and add bellow code :---

@Autowired

**private** CustomerUtil util;

//1.show Reg page

@RequestMapping("/custReg")

**public** String showPg(ModelMap map)

{

util.addToUi(map); //Also add in Register(Insert) method

**return** "CustomerReg";

}

**4>Add Bellow code in “CustomerController”:--**

Location : <select name=*"loc.locId"*>

<option value=*"-1"*>-- Select --</option>

<!-- locsui name must be same as CustomerUtil attribute variable of loc filed -->

<c:forEach items=*"*${locsUi}*"* var=*"loc"*>

<option value=*"*${loc.locId}*"*> ${loc.locName}</option>

</c:forEach>

</select>